--- PREPUBLICATION DRAFT---

WORKSHOP ON THE ROADMAP FOR THE

REVITALIZATION OF HIGH-END COMPUTING

June 16-20, 2003

Edited by

Daniel A. Reed

--- PREPUBLICATION DRAFT---

COMPUTING RESEARCH ASSOCIATION

Inside Front Cover

This workshop was sponsored by the National Coordination Office for Information Technology Research and Development (www.nitrd.gov).

The views expressed in this report are those of the individual participants and are not necessarily those of their respective organizations or the workshop sponsor.

This is a Prepublication Draft only. To request a copy of the final report, send an e-mail to: info@cra.org or call 202-234-2111.

TABLE OF CONTENTS

The most constant difficulty in contriving the engine has arisen from the desire to reduce the time in which the calculations were executed to the shortest which is possible.
Charles Babbage, 1791-1871

PREFACE

Several recent developments in high-end computing (HEC) have stimulated a re-examination of current U.S. policies and approaches. These developments include: 1) the deployment of Japan's Earth System Simulator, which now occupies the number one position on the Top 500 list of the world's fastest computers; 2) concerns about the difficulty in achieving substantial fractions of peak hardware computational performance on high-end systems; and 3) the ongoing complexity of developing, debugging, and optimizing applications for high-end systems. In addition, there is growing recognition that a new set of scientific and engineering discoveries could be catalyzed by access to very-large-scale computer systems—those in the 100 teraflop to petaflop range. Lastly, the need for high-end systems in support of national defense has led to new interest in high-end computing research, development, and procurement.

In recognition of these developments, the FY03 federal budget included the following observations and guidance regarding high-end computing:

> *Due to its impact on a wide range of federal agency missions ranging from national security and defense to basic science, high-end computing—or supercomputing—capability is becoming increasingly critical. Through the course of 2003, agencies involved in developing or using high-end computing will be engaged in planning activities to guide future investments in this area, coordinated through the NSTC. The activities will include the development of an interagency R&D roadmap for high-end computing core technologies, a federal high-end computing capacity and accessibility improvement plan, and a discussion of issues (along with recommendations where applicable) relating to federal procurement of high-end computing systems. The knowledge gained for this process will be used to guide future investments in this area. Research and software to support high-end computing will provide a foundation for future federal R&D by improving the effectiveness of core technologies on which next-generation high-end computing systems will rely.*

In response to this guidance, the White House Office of Science and Technology Policy (OSTP), in coordination with the National Science and Technology Council, commissioned the creation of the interagency High End Computing Revitalization Task Force (HECRTF). The interagency HECRTF was charged to develop a five-year plan to guide future federal investments in high-end computing.

To ensure broad input from the national community during its planning process, the HECRTF solicited public comments on current challenges and opportunities. Using the White Papers from this public request for comments, a program committee (see Appendix A) organized a community input workshop for focused discussions. Held June 16-18, 2003, the *Workshop on the Roadmap for the Revitalization of High-End Computing* was structured around eight working groups. Each group was given a specific and focused charge (summarized in Appendix B) that addressed a specific aspect of high-end computing (e.g., technology, COTS-based or custom architecture, software, and applications).

This report is a summary of the workshop findings. Its purpose is to provide guidance to the national community and the HECRTF when planning future national programs for the development, specification, procurement, deployment, and application of future high-end computing systems.

Acknowledgments
The report and the workshop would not have been possible without the diligence and hard work of a great many individuals. As workshop chair, I am indebted to the members of the program committee for their insights, experience and guidance. I am also thankful for the enthusiasm and response of the working group chairs and vice chairs. Without their coordination and writing, this report would not have been possible.

Lastly, the workshop benefited from the able assistance of the staff from the National Coordination Office (NCO) and the Computing Research Association (CRA). I am especially grateful to David Nelson (NCO), Thomas Sterling (Caltech/JPL), and Robert Sugar (University of California at Santa Barbara) for their work in developing the working group charges and questions.

Dan Reed, Workshop Chair
National Center for Supercomputing Applications
Urbana, Illinois

1. EXECUTIVE SUMMARY

Based on their deliberations, the workshop's eight working groups developed a set of key findings and recommendations to advance the state of high-end computing in the United States. The common theme throughout these recommendations is the need for sustained investment in research, development, and system acquisition. This sustained approach also requires deep collaboration among academic researchers, government laboratories, industrial laboratories, and computer vendors.

Simply put, short-term strategies and one-time crash programs are unlikely to develop the technology pipelines and new approaches needed to realize the petascale computing systems needed by a range of scientific, defense, and national security applications. Rather, multiple cycles of advanced research and development, followed by large-scale prototyping and product development, will be required to develop systems that can consistently achieve a high fraction of their peak performance on critical applications, while also being easier to program and operate reliably. A summary of the most critical findings of the workshop in each of the eight areas follows.

1.1. Enabling Technologies

Power management and interconnection performance are of great and mounting concern. Today's very-large-scale systems consume megawatts of power, with concomitant packaging and cooling problems. Equally important, many computations are limited by the bandwidth and latency among chips, boards, and chassis. A variety of new device technologies and three-dimensional integration and packaging concepts show promise in ameliorating the interconnect bandwidth and heat dissipation problems. We urge that attention be focused on these to move them toward system feasibility and prototype.

Looking further ahead, there are many novel devices based on superconducting technologies, spintronics, photonic switching, and molecular electronics that are exciting and can be game-changers for the high end. However, it is much too early to choose just a few, and we urge a broad, consistent, long-term research agenda to incubate and test these ideas in preparation for future generations of systems.

Software for large-scale systems requires scaling demonstration to assess its importance. New approaches that reduce time-to-solution through better shared computing models and libraries, coupled with real time performance monitoring and feedback, are needed to achieve the promise of high-end computing.

1.2. COTS-Based Architectures

The bandwidth of today's COTS memory systems and interconnection networks limits the performance of HEC applications. The advent of memory-class ports into microprocessors will permit implementation of higher-bandwidth, lower-latency interconnects. Moreover, higher-speed signaling and higher radix routers will make interconnection networks with higher bandwidths and flatter topologies practical. Finally, field programmable gate arrays (FPGAs) will embed specialized functions, application kernels, or alternate execution models (e.g., fine-grain multi-threading) in a format that is optimized yet malleable. However, realizing these advanced technologies will require continued investment.

More generally, we must develop a government-wide, coordinated method for influencing vendors. The HEC influence on COTS components is small, but it can be maximized by engaging vendors on approaches and ideas five years or more before the resulting commercial products are created. Given these time scales, the engagement must also be focused and sustained.

We must fund long-term research on the key technologies needed for future high-end systems. Beyond the research funding itself, academic researchers need access to HEC-scale systems to test new ideas in

realistic situations. Once ideas are proven in a research setting, the best ones must be incorporated into production systems by an intentional process.

1.3. Custom Architectures

Custom high-end computer architectures are designed explicitly to be incorporated in highly scalable system structures and operate cooperatively on shared parallel computation to deliver substantially higher performance, efficiency, and programmability than COTS-based systems, while imposing lower cost, space, and power requirements. Multiple technical approaches for custom-enabled architectures of significant promise have been identified that, with necessary funding, can be realized in the near and medium term, including, but not limited to: a) spatially direct-mapped architecture,(b) vectors, c) streaming architecture, d) processor-in-memory architecture, and e) special purpose devices.

Proof of concept of such innovative architectures is feasible within the next five years, and petaflops-scale computer systems can be deployed substantially before the end of this decade. However, to meet this objective it is imperative that research in advanced, custom scalable HEC architecture be sponsored at an accelerated and continuous level to regain U.S. leadership in this technology that is strategically critical for national security and commerce. New partnerships are required among industry, universities, government laboratories, and mission agencies.

Both system software and programming environments must be developed that support and exploit the capabilities of the custom architectures. System software must be developed to provide the dynamic resource management anticipated by many of these architectures in order to improve performance efficiency and remove much of the burden from the programmers. Programming environments must be developed that capture and expose intrinsic algorithm parallelism for greater performance and provide high-level constructs to eliminate low-level and error-prone detail to minimize application development time. In addition, effective software means must be provided to enable rapid porting of legacy applications and libraries to maintain continuity of the user software base.

1.4. Runtime and Operating Systems

Unix and its variants have been the operating system of choice for the technical world for nearly thirty years. While Unix has served the community well, its very design point and set of assumptions are increasingly at odds with the needs of high-end computing. Alternate resource management models are needed that provide better performance feedback for dynamic adaptation, including increasing coupling among operating system, runtime, and applications. New models for I/O coordination and security are needed as well. Simply put, more system software research that is revolutionary, rather than evolutionary, will be needed to support the next generation of petascale systems.

Lastly, the current lack of large-scale testbeds is limiting operating system and runtime research for the HEC community. Such testbeds can provide the broad research community with access to flexible testing environments and scalability research platforms. These testbeds should be available to university groups, national laboratories, and commercial software developers.

1.5. Programming Environments and Tools

The most pressing scientific challenges will require application solutions that are multidisciplinary and multi-scale. In turn, these software systems will require an interdisciplinary team of scientists and software specialists to design, manage, and maintain them. This will require a dramatically higher investment in improving the quality, availability, and usability of the software tools used throughout an application's lifecycle.

The strategy for accomplishing these goals is not complex, but it requires an attitude change about software funding for HEC. Software is a major cost component of modern technologies, but the tradition

in HEC system procurement is to assume that the software is free. Mission critical and basic research HEC software is not provided by industry because the market is so small and the customers are not willing to pay for it. Federally funded management and coordination of the development of high-end software tools for high-end systems are needed.

Funding is also needed for basic research and software prototypes, and for the technology transfer required to move successful prototypes into real production quality software. It is urgent that we invest in building interoperable libraries and software component and application frameworks that simplify the development of these complex HEC applications. It is also essential that we invest in new, high-level programming models for HEC software developers that will improve productivity and create new research programs that explore the hardware/software boundary to improve HEC application performance.

Structural changes are needed in the way funding is awarded to support sustained engineering. We need a software capitalization program that resembles the private sector in its understanding of the software lifecycle. One approach to coordinating a federal effort in this area would be to establish an institute for HEC advanced software development and support, which could be a cooperative effort among industry, laboratories, and universities.

1.6. Performance Analysis

The single most relevant metric for high-end system performance is *time to solution* for the specific scientific applications of interest. Reducing the time to solution will require aggressive investment in understanding all aspects of the program development and execution process (programming, job setup, batch queue, execution, I/O, system processing, and post-processing).

The current practice in system procurements is to require vendors to provide performance results on some standard industry benchmarks and several scientific applications typical of those at the procuring site. Constructing these application benchmarks is a cost- and labor-intensive process, and responding to these solicitations is very costly for prospective vendors. Moreover, these conventional approaches to benchmarking will not be suitable for future acquisitions, where the system to be procured may be over ten times more powerful than existing systems.

Recent successes with performance modeling suggest that it may be possible to accurately predict the performance of a future system, much larger than systems currently in use, on a scientific application much larger than any currently being run. However, significant research is needed to make these methods usable by non-experts. Research is also needed to bolster capabilities to monitor and analyze the exploding volume of performance data that will be produced in future systems. All of this research will require significant involvement by vendors, and thus some dialogue will be needed to resolve potential intellectual property issues.

1.7. Application-Driven System Requirements

Multiple disciplines made the quantitative case for speedups in sustained performance of 50 to 100 over current levels to reach new, important scientific thresholds. In QCD, architectures with a sustained performance of 20 to 100 TF would enable calculations of sufficient precision to serve as predictions for ongoing and planned experiments. In magnetic fusion research, sustained execution of 20 TF would allow full-scale tokamak simulations that resolve the natural length scales of the microturbulence, as well as enable self-consistent gyrokinetic modeling of the critical plasma-edge region. Lastly, 50 TF was identified as an important threshold for developing realistic models of lanthanides and actinides on complex mineral surfaces for environmental remediation, and for developing new catalysts that are more energy efficient and generate less pollution.

Applications have become so complex that multidisciplinary teams of application and computer scientists are needed to build and maintain them. The traditional software model of a single programmer developing a monolithic code is no longer relevant at the high end and cutting edge. In particular, large teams with diverse domain expertise are needed to integrate multi-scale simulation models. No single person or small group has the requisite expertise.

Application codes and their associated analysis tools are the instruments of computational science. The developers of these applications can be likened to instrument specialists in that they possess the most detailed knowledge of the applications' capabilities and usage. Unlike experimental science, however, the computational end stations need not be located at the HEC facilities. Within this facilities analogy, HEC users in the form of collaborative research teams would interface primarily with the application specialists within domain-specific research networks that develop, optimize, and maintain the relevant applications.

1.8. Procurement, Accessibility, and Cost of Ownership

Procurements should use functional specifications to define science requirements and the application environment. One should also minimize the number of mandatory requirements, avoid restricting competition, and permit flexible delivery schedules. The total cost of ownership, together with technical and risk assessments, should be the primary evaluation criterion. Agencies that are large users should provide HEC services to agencies that are small users employing any of the appropriate, standard interagency agreement vehicles, and the using agency must supply a *multi-year financial commitment* to the supplying agency.

2. ENABLING TECHNOLOGIES FOR HIGH-END COMPUTING

Sheila Vaidya, Chair
Lawrence Livermore National Laboratory

Stuart Feldman, Vice Chair
International Business Machines (IBM)

There are two distinct approaches to achieving the performance increases needed by future high-end computer systems: 1) highly scalable architectures exploiting many-million-way parallelism, and 2) advanced component technologies with dramatic improvements in critical properties. Reliance on new or improved technologies to provide significant gains in system performance has been a major strategy throughout the half-century history of digital computer development. Clock rates of 10 KHz in 1950 have increased to 1 GHz in 2000, five orders of magnitude or a tenfold increase every decade. Even as the first strategy of unprecedented parallelism is used to deliver operational capability in the trans-petaflops performance regime, advanced technologies are required to make such unique system architectures both feasible to build and practical to operate. In addition to affecting logic switching speed, technology advances impact memory density and cycle time, logic density, communication bandwidth and latency, power consumption, reliability, and cost.

Advances include both incremental improvements of existing device types (e.g., embedded DRAMs) and novel technologies for new component classes (e.g., MRAMs or holographic storage). In extreme cases, technology advances may inaugurate new computing paradigms (e.g., quantum computing). The working group considered basic technology and components, both hardware and software. In particular, it addressed areas that are unique to high-end computing or are unusually stressed by it, and that would not be delivered by the commercial sector in time to meet the needs of the high-end computing community. While not exhaustive, several promising opportunities, technical strategies, and challenges to the development and exploitation of possible future device and component technologies are described.

2.1. Enabling Technologies

2.1.1. Device Technologies

Semiconductor. Essentially all current computing systems are based on complementary metal oxide semiconductor (CMOS) integrated circuits, and billion-transistor chips will soon be routinely available. Such progress has been made possible by continuing technological improvements in semiconductor fabrication: advanced lithography, copper interconnects, and low-k dielectrics, to name a few. The International Technology Roadmap for Semiconductors (ITRS) [15] projects that feature sizes will scale down to 22nm by 2016. However, key technological enhancements must be emphasized in the near term for added benefit to high-end computing.

Silicon-on-Insulator (SOI) CMOS promises lower power integrated circuits. Although there is industrial interest in this technology for mobile applications, an aggressive scaling of high-performance SOI-based System-on-Chip (SoC) could prove extremely valuable to HEC as the chip count/package and, hence, total power dissipated/node, rises. Simultaneously, device technologies that enable SoC Processor-In-Memory (PIM) architectures must be supported aggressively. These include smart memories and small-scale, multithreaded multiprocessor concepts, as well as novel three-dimensional device constructs. The latter will allow building upward on a CMOS SoC underlayer using advanced processing concepts such as laser annealing.

Furthermore, integration of electro-optical components (based on III-V compounds) on silicon—specifically, on-chip lasers for driving optical signals—coupled with fast, intelligent (all-optical) routers

can be extremely beneficial in building integrated, low-latency backplanes. A concerted research and development effort in this area, which incorporates advanced three-dimensional packaging concepts, could significantly benefit the design of high-end computing systems.

Lastly, because CMOS performance improves at lower temperatures (~77°K) due to both higher switching speed and lower power dissipation, a variety of low-temperature approaches must be evaluated from a cost/performance perspective for possible implementation in the medium term. Several practical problems related to integration of refrigeration and maintenance duty cycles must be demonstrated in production prototypes, but the opportunities could be equally significant for high-end computing.

Superconducting Technologies. Superconducting technologies exploit the physical property that some materials exhibit zero electrical resistance when cooled below a critical threshold temperature. For niobium on a silicon substrate, this critical temperature is approximately 4°K. Essentially all superconductor logic devices are based on the Josephson Junction (JJ), a current-driven switching device projecting two states: a zero resistance state and a high resistance state. Although early work proved disappointing, a new class of superconductor logic based on the "squid" (a loop of two Josephson Junctions and an inductor), originally developed for highly sensitive sensors, has yielded dramatic improvements in logic speed and low power.

Rapid Single Flux Quantum (RSFQ) gates have been demonstrated with switching rates exceeding 700 GHz. Although today's manufacturing processes at a micron resolution deliver clock rates in the 20 to 40 GHz range, new sub-micron process technologies could enable 100 and 200 GHz clocks with power consumption of 0.1 microwatt per gate. Development of these advanced fabrication procedures, as well as accompanying logic and memory architectures, should be the basis of future, mid-term research and development. (A previous point design study, HTMT, demonstrated that a petaflops-scale computer system based on RSFQ logic would require only 400 square feet of floor space and consume less than one megawatt of power, including cooling.)

Nanotechnology. In the longer term, beyond the fundamental limits of silicon scaling, radical changes in materials and processes used for component fabrication may be required. Several nanotechnologies show promise for improving areal density, data communication, and 3-D assembly. These could enhance parallel memory access for logic operations, and provide high interchip bandwidth or inter-board communication as well. MEMS (Micro-Electrical Mechanical Systems) are one such enabling technology for improving data storage via high areal density, massive parallelism, and lower power consumption, albeit with slow response. Researchers believe that 1 terabit/square inch is easily achievable with MEMS, about 30 times that of current magnetic hard drives.

MEMS storage devices are inherently parallel by nature of the optical read/write process. In addition, MEMS, judiciously deployed, could enable high-bandwidth, free-space, optical communication among compute nodes. While MEMS-based optical routing is being explored for long haul communications, system design studies and tradeoffs for high-end applications are currently lacking.

Spintronics, the development of devices that harness the spin degree of freedom in materials, not just their charge, is another important research and development area with a payoff of roughly ten years. A current embodiment of spintronics is the magnetic random access memory (MRAM), which is being commercialized by Motorola and IBM/INFINEON as a replacement for FLASH memory. MRAM is nonvolatile, has the potential of matching SRAM speeds at DRAM densities, and scales favorably with design rule. The device consists of two electrically conducting magnetic materials separated by an insulator. When the layer magnetizations are parallel, the device impedance is low; when they are anti-parallel, the impedance is high, representing a "1" or a "0" bit.

The use of MRAMs, coupled with CMOS logic in creative configurations to mitigate the memory performance bottleneck, could have an immediate payoff for the HEC community. In the longer term, alternate spin-based transistors and light-emitting devices, which are the subject of sporadic university research in the United States today, could provide novel micro-architectural building blocks for future HEC.

Alternate nanodevices based on molecular electronics are also being explored at places such as HP Laboratories to evolve novel fault-tolerant HEC platforms. Some of these approaches could potentially enable entirely new paradigms in switching, memory and system reliability. We urge an increased focus on these areas in order to understand their capabilities and limitations in a timely manner, since they could revolutionize "conventional" design concepts in supercomputing platforms.

2.1.2. Memory Devices

In the very near term, custom silicon integrated circuits for intelligent, adaptable memory systems—with accompanying middleware and operating system to manage them and compiler technology and performance tools to exploit them—could be a valuable asset to both uniprocessor and PIM organizations, as well as shared memory multiprocessor nodes. Although prototype intelligent memory systems are being considered (e.g., the Impulse memory controller), they fall short in coherency management and synchronization control. The design and fabrication of such custom hardware is impeded by poor profit margins on such low-volume components. Hence, the need for customized memory-controllers underscores a larger problem of making available custom integrated circuits/packages to test possible design innovations for HEC. Without some means of cost-sharing this expense with the private sector, progress in optimizing SoC architectures for HEC will be slow and sporadic.

High-end computing demands low-latency, high-bandwidth access to main memory for many of its applications. While latency tolerance can be built into the SoC, memory bandwidth remains by far the most important technological hurdle facing high-end subsystems today. Hence, for mid-term impact on HEC, it is necessary to aggressively pursue concepts that: 1) enlarge memory capacity on-chip, in close proximity to the arithmetic operations (such as three-dimensional integration of memory on CMOS);
2) provide high-speed access off-chip (waveguide/free space optical interconnects coupled with high-bandwidth WDM-based smart network topologies); and 3) deliver three-dimensional storage and readout from main memory (such as MEMS or holographic storage).

2.1.3. Storage and I/O

For some mission applications, the critical resource and performance driver is not the peak floating point performance or even the main memory bandwidth, but rather the capacity, speed, and logical accessibility of massive amounts of data. Data sizes of many petabytes, even approaching exabytes, may be critical for some tasks. Cost-effective storage and rapid access, as well as reliable storage and communication, will dominate the quality, utility and cost at many deployed sites. Even today, the major cost of many systems is for the secondary storage system rather than the processors. Advanced technologies, which include software, must be developed over the period of consideration to meet the exponentially increasing demands for effective data storage and management.

Advanced technologies for low-cost mass storage that have demonstrated promise and should be aggressively pursued in the mid term include holographic optical storage using photo-refractive components and spectral hole burning, MEMS, scanning tunneling microscopy and other e-beam-based techniques, and three-dimensional magnetic storage using MRAMs. In the long term, molecular electronics may hold the key to the ultimate in storage density. In the near term, however, better I/O controllers and remote DMA facilities, in conjunction with improved software technologies for cluster storage access and higher-level object-based storage systems, could prove extremely invaluable.

2.1.4. Interconnects

Off-chip interconnection for high-end systems remains our greatest concern. Even today, off-chip electronic connections are inadequate because they degrade chip clock frequencies by an order of magnitude or more. Coupled with the ramifications of DRAM speed and memory organization as well as conventional microprocessor cache hierarchy, sustained system performance of 1 to 5 percent of peak is not uncommon in applications that require high global/local communication bandwidth. As our appetite for bandwidth between memory and microprocessors grows, it is very likely that new technologies (e.g., optical communication links with smart network topologies that balance average bandwidth against number of links) will become mandatory. The following section separates interconnect technology into two categories and discusses them individually.

Passive Interconnects. Evolution in optical network technology for metropolitan area networks is forcing significant changes in both the active and passive components of communication links. By definition, passive interconnects deploy fixed elements, which implies minimum dynamic network optimization or error correction capability. Conversely, active interconnects imply real-time software control, integrated with the loop architecture, to ensure optimum operation with respect to each of the codependent active components.

A concerted effort to leverage the telecommunication infrastructure for high-end interconnections should include evolving a high-speed serial optical interface for fiber backplanes in the near term, and high-speed photonic switching in the long term. In the mid term, reliable wavelength division multiplexing (WDM) fiber or free-space optical interconnects for board or inter-board communication—interconnecting terascale node boards with packet-switched, source-routed communication links—can have a tremendous payoff for high-end platform design. This development should be encouraged, while simultaneously accelerating component research and development activity from universities into mainstream industry for system implementation.

Beyond ten years, on-chip optical waveguide clocks are plausible. Again, while such research activity falls under the purview of basic science, without additional investment of resources such enhancements could remain an unfulfilled dream for the high-end computing community.

Active Interconnects. The shift from passive to active optical components holds great promise for increasing network flexibility and robustness. Dynamic wavelength rerouting can provide operational efficiency and protect against performance degradation. Active modules can also suppress transients and manage power fluctuations. Furthermore, incorporating intelligence into the network architecture and using active filters with tunable sources can increase both flexibility and bandwidth; active compensation of filtering response can allow for tighter channel spacing.

However, the shift from passive to active interconnects comes with an incremental cost in research and development that the telecom industry can least afford today. Hence, the adoption of active networking concepts into the mainstream communication environment remains slow. Therefore, it is incumbent on the government to aid in screening and evolving those active interconnect technologies and to facilitate their transition to prototypes. Specifically, concepts involving intelligent electronic crossbar switches, dynamic network processing on-board, and data vortex constructs lend themselves to near- to mid-term payoffs, and should be aggressively pursued.

2.1.5. Single Chip Architecture

Emerging semiconductor technology and fabrication processes are integrating CMOS logic and DRAM on the same silicon semiconductor substrate. This capability enables a new family of hardware computing constructs referred to as “single chip architectures.” A System-on-Chip (SoC) integrates a conventional processor core with a complete memory hierarchy, including one or more levels of SRAM cache and one

or more banks of DRAM main memory. Symmetric multiprocessing (SMP) on a chip co-locates two or more conventional processor cores on the same semiconductor die with snooping mechanisms for cache coherence across L1 and L2 SRAM caches private to each processor, and an L3 DRAM cache shared among all processors. Processor-In-Memory (PIM) positions logic next to DRAM row buffers. This exposes all bits of a memory block (typically 2048 bits) to low-latency logic for in-memory processing. In addition, DRAM memory may be partitioned into multiple banks, each with its own wide logic array, greatly increasing the internal memory bandwidth.

Single chip architectures will play an important role in future high-end systems, providing smaller packages, reduced power consumption, low latency of access, higher logic density, higher effective bandwidth, and greater hardware parallelism. They can enable fine-grained, irregular parallel computing paradigms and reconfigurable circuits for enhanced system performance and reliability. However, future research and development will be required to improve the performance characteristics of the basic storage and logic devices, to increase density and decrease power consumption further, and to devise new architectures—perhaps adaptive, possibly deploying asynchronous logic—that can best exploit this new capability.

2.1.6. Power/Thermal Management and Packaging

Both users and vendors are increasingly concerned with reducing power consumption, decreasing footprints, and managing the heat generated in large-scale computing environments. Based on the ITRS roadmap [15], high-performance CMOS processors will generate 120 to 200 watts of heat by 2009. ASCI Purple, projected at 100 peak teraflops, is expected to consume 5 megawatts of power, plus an additional 3.5 megawatts for cooling and power management. Future high-end systems will co-locate hundreds of integrated circuit chips, running at higher clock rates, with RF, MEMS and other optical components. Thus, minimizing floor space and power are important imperatives not only for machines serving the national security community, but also for future servers and workstations that service the private sector. Research in novel heat dissipation and packaging methodologies must closely parallel developments in system architecture and device design.

Some commercial directions (e.g., lower chip voltages, dynamic power management on chip, and operating system optimization) are already being tapped. More aggressive cooling methodologies that deploy evaporative film boiling or spray cooling of devices with low-boiling-point inert fluids need to be exploited at the full-scale system level in the near term. Commercial manufacturers have avoided incorporating liquid cooling technologies due to the up-front financial burden of technology change as well as customer discomfort. However, a step function away from forced air cooling will be necessary as we invoke higher-density packaging (2.5-D initially, where a few layers of active components are stacked atop each other) for faster and shorter connectivity.

In the longer term, true 3-D packaging will be needed to achieve shorter latencies and enhanced raw computing capability. New concepts will be required, some of which could be borrowed from other applications. For example, micro-channel cooling, which relies on single phase liquid convection in silicon micro-channels, can dissipate heat fluxes on the order of kilowatts/cm^2. This technology has been lifetime-tested and is in use for cooling three-dimensional stacks of high-performance AlGaAs laser diode arrays. The concept could be adopted for high-end subsystems, coupled with active temperature control (i.e., dynamic management of device temperatures to monitor hot spots, which could be compensated for by changing local clock speeds and reallocating workloads). However, development resources will be necessary to assess the feasibility of these concepts, especially in light of the concomitant requirement of vertical interconnects for data communication through these packages.

Fundamental research is needed to progressively improve the computing power per watt criterion as systems grow in scale. Moreover, within ten years, the current semiconductor roadmap for operations/

watt flattens. Because the demand for total operations per second will continue to rise, breakthroughs in devices and packaging will be essential to deliver platforms with adequate manufacturability.

2.1.7. Software Methods and Algorithms

In addition to hardware technologies, new fundamental advances in software technologies are necessary to enable and exploit effective petascale high-end systems. Major advances are required in both application techniques and resource management software methodologies. Principal directions for future research and development in software technology for HEC are considered below. Other working groups also considered these issues in their discussions (see chapters 5 and 6).

Parallel Algorithms. Future petascale computers will require billion-way parallelism to achieve their performance goals. Hence, new classes of algorithms will be required that expose high degrees of parallelism, including intrinsic fine-grain parallelism, to the compiler and system hardware. In addition, the overhead for communication and synchronization must be sufficiently low to permit lightweight threads to function efficiently. If future high-end systems are to succeed, advanced algorithmic techniques must be devised that incorporate these properties for a wide range of important applications.

Runtime and Operating Systems. Operating system software has changed little over the past two decades, although research continues. Microprocessor-based massively parallel processors and clusters, combined with Unix and Linux, have defined the status quo for an extended period. New generations of high-end systems incorporating many millions of execution sites and distributed shared memory—and perhaps message-driven, lightweight, transaction processing with multithreading hardware support—will demand an entirely new class of resource management and task supervisor software.

Simple local support routines, synthesized into a single-system image, global master control program (MCP), may be based on the synergism of semi-independent agents that monitor and react to changes in system state and status employing methods like introspective threads. Such new environments would be highly robust because the failure of any one agent, and/or its underlying local hardware support, would not imply the failure of the entire system. Research is required to develop this new generation of system management and runtime software technology.

High-Level Languages and Compilation. Today's parallel computers are difficult to program using conventional models, techniques, and tools. Future many-million-way parallel petascale computers will become almost impossible to program without significant advances in programming methodology and technology. One aspect of this will be new, very-high-level formalisms or programming languages that facilitate the capture of the user-defined functions with a minimum of effort, while also exposing algorithm characteristics of parallelism and affinity (temporal and spatial locality). Such frameworks must enable the rapid integration and reuse of distinct program modules that were not necessarily written to form a single program.

To make this possible, extended methods like the common component architecture (CCA) discipline must be developed to guarantee computation that is easy and effective, cooperative and coordinated. Compilation strategies and tools must be devised that expose and manage both parallelism and affinity, while coordinating with the services provided by future runtime system software. Both future languages and compiler technology must include facilities for performance and correctness debugging.

2.2. Summary of Strategic Findings

From the wealth of facts and considerations derived from this important community forum, several key observations emerged that should be incorporated in any planning of future high-end system procurement and development. The most significant among these are presented below.

Areas currently raising concerns that are likely to remain challenging for many years to come are the management of power and improvements in interconnection performance. We urge considerable investment, in both the short and medium term, in these areas.

Seymour Cray famously stated that his main contributions to supercomputing were related to "plumbing." Managing heat and reducing power use continue to be major problems. Almost by definition, high-end computing calls for larger numbers of computing devices than more common installations, exacerbating traditional problems.

Many computations are limited by the bandwidth and latency among chips, boards, and chassis. Numerous forms of fast networking and interconnects have been examined, but most commercial computers do not yet require a shift to optical interconnects. We consider research and development in this area to be a very high priority. There are many types of computations that are fundamentally limited by interconnect delays because they cannot fit all of the necessary information into on-chip memory.

In addition, a variety of new device technologies and three-dimensional integration and packaging concepts show promise for ameliorating the problems with interconnect bandwidth and heat dissipation. We urge that attention be focused on these to advance them to the stage of system feasibility and prototype.

Looking further into the future, there are many novel devices based on superconducting technologies, spintronics, photonic switching, and molecular electronics that are exciting and can be game-changers for HEC. However, it is much too early to choose just a few, and we urge a broad, consistent, long-term research agenda to incubate and test these ideas in preparation for future generations of systems.

Some important software approaches require system demonstration to assess their importance. The purpose of HEC is to solve new computationally intensive problems, cost-effectively and expeditiously. Approaches that fundamentally address the time to solution through better shared computing models and libraries, coupled with real-time performance monitoring and feedback, are needed to achieve the promise of high-end computing.

2.3. Enabling Technology Roadmap

2.3.1. Time Scales and Investments

Radical technologies require both time and investment before they become practical for incorporation into major systems. We believe that one of the key roadblocks to greater progress in HEC capability has been the sporadic investment in its future. Therefore, we call for a continuous pipeline of research and development resources to nucleate new ideas and to move forward—from concept into product—those ideas that solve specific problems facing high-end computing,. A key point driving our conclusions was the practical time scale and costs involved in introducing radical, or even significantly changed, technologies into large-scale systems.

Because of the complexities of market dynamics and the limited demand for high-end computing, it is difficult for a new and enabling concept to appear in a full-scale, high-end system in less than 7 to 10 years, if at all. Hence, we recommend a focused parallel effort that explores technology relevance to other industrial applications to accelerate their early adoption into production prototypes. This could be enforced by, for example, including in our assessment application suite some complex aircraft design or automobile crash simulation codes that could benefit from optimized HEC capability. This would expand the market need and would help to leverage government investments with private sector funds as the systems became available.

In terms of nomenclature, we have assumed that any technology that will appear in full-scale deployments (large-scale systems in production use) by 2009 already exists and has been tested in subsystems, not just on the laboratory bench. Clearly, variations will be needed for integration into high-end computing. This is what we are calling out as a near-term investment need. The themes here are few, but they demand immediate attention. Mid term takes us into the next phase where concepts have been proven/invented and their usefulness to high-end computing established. However, system embodiments must be built and tested and their scaling implications must be assessed. These capabilities will only be available in high-end platforms in the next decade. Here, many more possibilities exist. These should be studied in pilot environments where cross interactions can be analyzed. This calls for additional resources.

Of course, several technologies are being explored that could enable novel systems approaches. It is far too early to pick winners amongst these. Instead, a broad set of initiatives should be funded with periodic checkpoints that can separate the likely winners in key areas.

2.3.2. Key Technologies

Based on working group discussions, Table 2.1 summarizes the group's assessment of technology evolution and development during the next fifteen years.

FY05-FY09	FY10-FY14	FY15-FY19
Devices		
Silicon on Insulator	Low temperature CMOS	Nanotechnologies
Si-Ge	Superconducting RSFQ	Spintronics
Mixed III-V devices		
Integrated electro-optic and high speed electronics		
Memory		
Optimized memory hierarchy	3-D memory (e.g., MRAM)	Nanodevices
Smart memory controllers		Molecular electronics
Storage and I/O		
Object-based storage	Software for cluster storage access	Spectral hole burning
Remote DMA	MRAM, holographic, MEMS, STM and E-beam	Molecular electronics
I/O controllers (MPI, etc.)		
Interconnects		
Optical system area networks (fiber-based)	Active networks	Scalability (node density and bandwidth)
Serial optical interface	High-density optical networking	
Electronic crossbar switch	Optical packet switching	
Network processing on board	Data vortex	
	Superconducting crossbar switches	
Single Chip Architectures		
Power efficient designs	Adaptive architecture	
System-on-Chip (SoC)	Optical clock distribution	
Processor-in-Memory (PIM) (e.g., Caltech MIMD)	Asynchronous designs	
Reconfigurable circuits		
Fine-grained irregular parallel computing		
Packaging and Power		
Optimization for power efficiency	3-D packaging and cooling (e.g., microchannel)	Higher scalability concepts (improving operations/watt)
2.5-D packaging	Active temperature control	
Liquid cooling (e.g., spray)		
Software and Algorithms		
Compiler innovations for new architectures	Very high level language hardware support	
Tools for robustness (e.g., delay and fault tolerance)	Real time performance monitoring and feedback	
Low overhead coordination mechanisms	Parallel Random Access Machine model (PRAM)	
Performance monitors		
Sparse matrix innovations		

Table 2.1 Enabling Technology Opportunities

3. COTS-BASED ARCHITECTURES

Walt Brooks, Chair
NASA Ames Research Center

Steven Reinhardt, Vice Chair
SGI

Commodity off-the-shelf (COTS) components are currently used to construct a wide variety of commercial and custom high-end computing systems. The working group's charter focused on COTS-based systems, rather than on systems composed solely of COTS components. Based on this, we defined COTS in three overlapping but non-identical ways. The first definition considered technologies intended for enterprise or individual use (e.g., commodity processors, memories, and disks). The second considered technologies that require such a large investment to replicate that the high-end community must exhaust every possibility of using the commodity components before building specialized components for high-end systems. The third considered technologies whose evolution is largely determined by other markets; the HEC market has little influence over these. The working group's bias was that application needs must be met by the systems delivered (whether COTS-based or custom), and that COTS deficiencies be viewed as challenges to be overcome rather than be reflected as systems with constrained capabilities.

We considered the capability roadmap of the anticipated COTS-based HEC system architectures through the end of the decade. This roadmap identified the critical hardware and software technologies and architecture developments required, both to sustain continued growth and to enhance user support. Although current COTS-based systems are well suited to a variety of important problems, the group concentrated on their shortcomings for addressing key problems of national interest.

3.1. Enabling Technologies

Current COTS-based systems typically contain a small non-COTS component, often limited to the network interconnect (e.g., Myrinet or Quadrics). Although these systems are suitable for a variety of tasks, the working group believes that COTS-based systems can address a wide range of application needs by the inclusion of existing or imminent technologies. Our vision is of a computing fabric incorporating a heterogeneous set of computing devices. In addition to COTS-based processors and their associated memories, we expect COTS-based systems to include improved interconnects and support for reconfigurable computing.

The advent of memory-class ports into microprocessors will permit implementation of higher-bandwidth, lower-latency interconnects. Coupled with other technology advances (e.g., higher-speed signaling and higher-radix routers), interconnection networks with higher bandwidths and flatter topologies will be practical. However, as the working group addressing enabling technology noted in chapter 2, realizing these advances will require continued investment.

Initially implemented via field programmable gate arrays (FPGAs), reconfigurable or application-specific computing will embed specialized functions, application kernels, or alternate execution models (e.g., fine-grain multi-threading) in an optimized yet malleable form. Moreover, we believe functions with broad appeal to HEC applications (e.g., global address space or collective operations) will be implemented in system ASICs (application specific integrated circuits).

Based on working group discussions, these and allied topics are discussed below.

3.1.1. Memory Systems and Interconnects

We must increase memory bandwidth and provide mechanisms for using it more effectively. High-end computing applications are typically more demanding of memory bandwidth than the commercial workloads that are the optimization target for COTS processors. Simply put, the memory bandwidth on current and projected COTS processors is not increasing fast enough.

Despite the economic realities, the HEC community should influence the vendors of COTS processors to provide higher-memory bandwidth, recognizing that changes will take three to five years to appear in commercial products. Outside the processor itself, we should investigate new computational structures that exploit scarce processor-memory bandwidth as effectively as possible. Examples include moving specific functions (e.g., reductions or fast Fourier transforms (FFTs)) into ASICs or FPGAs.

Like memory systems, the bandwidth and latency of today's interconnection networks limit application performance. To redress this problem, two approaches are required. First, we must increase the bandwidth of links and routers and use higher radices in routers. The bandwidths required will not be funded by the non-HEC commercial market, but can be developed by HEC-focused vendors.

The second approach must exploit memory-class mechanisms for processor communication with remote nodes. This is in contrast to current commodity I/O-class mechanisms, whose latency and bandwidth do not support HEC needs. Fortunately, some recently developed COTS processors use point-to-point links for memory connection. (AMD's Opteron processor and its HyperTransport links are an example of this node architecture, with similar abilities expected from other COTS vendors.) These provide the needed ports and a new opportunity for system architecture design.

3.1.2. System Heterogeneity

HEC applications span a broad spectrum with a diverse set of resource demands. Because matching each portion of an application to its execution target (ASICs, FPGAs, or multiple processor types) can yield better overall performance, we must provide architectures that support heterogeneous elements in the system fabric. ASICs implement performance-critical functions that are used by numerous applications.

FPGAs can be reconfigured to accelerate those portions of application code that have the greatest computational intensity or need special functions. However, programming tools for FPGAs must improve greatly if FPGA use is to become more widespread. A multiplicative benefit of supporting FPGAs is that the ability to rapidly reconfigure functionality could accelerate the rate of HEC architectural research. Lastly, support for multiple processor types would allow HEC-optimized processors to perform I/O through COTS processors and their mass-market I/O adapters.

3.2. Summary of Strategic Findings

Two critical findings emerged from the working group discussions. We believe their implementation could accelerate and enhance the use of COTS-based technologies in high-end systems.

3.2.1. Government Coordination

We must develop a government-wide, coordinated method for influencing vendors. The HEC influence on COTS components is small, but it can be maximized by engaging vendors on approaches and ideas five years or more before commercial products are created. Given these time scales, the engagement must also be focused and sustained.

We recommend that academic and government HEC groups collect and prioritize a list of requested HEC-specific changes for COTS components, focusing on an achievable set. This process should also investigate commercial needs to identify overlaps with HEC; this will increase the alignment between the

COTS manufacturers and HEC needs. To the extent that HEC needs do not overlap with commercial needs, the government should fund the COTS vendors to make these changes.

Vendors of COTS-based HEC systems can address some of HEC's key technology needs in their systems. They may be more easily influenced than the COTS vendors themselves. Again, early focused government funding will likely be necessary to ensure robust implementations of innovative technology.

3.2.2. Long-Term Research Funding

We must fund long-term research on the key technologies needed for future high-end systems. We expect the current situation, where COTS components often meet HEC's key requirements poorly, to persist indefinitely. To combat this, a comprehensive research agenda is needed to mitigate component shortcomings and develop those key technologies that could be incorporated by the COTS manufacturers. Beyond the research funding itself, academic researchers need access to HEC-scale systems to test new ideas in realistic situations. Once ideas are proven in a research setting, the best ones must be incorporated into production systems by an intentional process, such as vendor funding discussed earlier.

Whereas development-stage actions are crucial, government must continue its role as the primary early adopter for high-end systems. This step is essential to reduce the risk for COTS-based vendors, as the capital cost of the largest high-end systems will not be borne by the vendors. The government should not only procure the systems, but also actively use the innovations to address problems of agency interest in order to understand their utility.

3.3. Other Challenges

3.3.1. Parallel Processing

Today's COTS processors have inadequate support for parallel processing. Although a few key shortcomings could be addressed by the recommendations above, we believe a more strategic approach would be to diffuse the use of parallel processing across the broad computing market. This step has the advantages of providing the benefits of parallelism to a greater audience, increasing the economic benefit to the nation, causing the mainstream markets on which the COTS manufacturers concentrate to be better aligned with the needs of high-end computing.

While this statement is simple, its implementation is not. Current methods commonly used for parallel programming are not suitable for use by non-experts. Much simpler interfaces for using parallelism must be devised. Without the motivation and expertise of the high-end computing community, this is unlikely to happen. This indirect influence on the COTS vendors can be strategic, but it is likely to take ten to fifteen years to bear fruit.

3.3.2. Scalable Software and Tools

There are also several areas where work is required to increase the scalability of software and tools needed for efficient use of very-large-scale systems. Linux must scale to support systems with 10,000 processors, and file systems also scale to support systems with exabytes (10^{18}) of storage. Similarly, compilers are needed that can generate efficient code for the heterogeneous architectures described above. Lastly, programming interfaces with lower overhead than MPI must be devised to enable scaling very large numbers of processors.

4. CUSTOM ARCHITECTURES

Peter Kogge, Chair
University of Notre Dame

Thomas Sterling, Vice Chair
California Institute of Technology

As others have noted, there are two strategies for achieving the high-end computer systems of the future: COTS-based systems and custom-enabled computer architecture. COTS-based parallel system architectures exploit the development cost and lead-time benefits of incorporating components—including microprocessors, DRAM, and interface controllers developed for the mainstream computing market—in highly replicated system configurations. The working group considered the opportunities, technical strategies, and challenges to realizing effective computing performance across the trans-petaflops regime through possible custom-enabled, high-end computer architectures.

Custom-enabled architectures are designed expressly for integration in scalable parallel structures to deliver substantially higher performance, efficiency, and programmability than COTS-based systems, while requiring lower power and less space. Both approaches are likely to lead to petascale performance before the end of this decade, but may exhibit very different operational properties as they are deployed and applied to compute and data-intensive applications critical to national security and commerce.

4.1. Custom-Enabled Architecture

A custom, high-end computer architecture is one that has many of the following characteristics:

- Its major components are designed explicitly to be incorporated in highly scalable system structures, and operate cooperatively on shared parallel computation to deliver high capability, short time to solution, and ease of programming.
- It is balanced with respect to rate of computing, memory capacity, and network communication bandwidth.
- It exploits performance opportunities afforded by device technologies through innovative structures that are not taken advantage of by conventional microprocessors and memory devices.
- It incorporates special hardware mechanisms that address sources of performance degradation typical of conventional architectures, including latency, contention, overhead, and starvation.
- It supports improved parallel execution models and assumes more responsibilities for global management of concurrent tasks and parallel resources, significantly simplifying programmability and enhancing user productivity.

Even though specialized devices are key to the success of the strategy of custom architectures, COTS components are and should be employed where useful when performance is not unduly sacrificed.

There is a wide range of possible custom parallel architectures, varying both in strategy and generality. In all cases, the objectives of their development are to:

- enable the solution of problems we cannot solve now, or of much larger versions of problems that we are currently solving on conventional COTS-based systems through dramatic capability improvement;
- deliver performance that is orders of magnitude better with respect to cost, size, and power than contemporary COTS systems at the performance scale for which they were designed; and

- significantly reduce the time to solution through both execution performance and enhanced programmability.

4.1.1. Strategic Benefits

By their nature, custom architectures promote a diversity of architecture by relaxing the constraints of system design imposed by conventional COTS microprocessors, and open opportunities for either alternative or point-design solutions to high-end computing problems that are far more efficient than currently possible. Their peak operation throughput and internal communication bandwidth for a given scale system may exceed equivalent attributes of conventional systems by one to two orders of magnitude, overcoming what are often referred to as the roadblocks for current technology. Overall system efficiency may be increased by up to an order of magnitude or more for some challenging classes of applications by means of hardware mechanisms devised expressly for efficient control of parallel actions and resources.

Enhanced programmability is a product of reduced barriers to performance tuning and elimination of many sources of errors, thus simplifying debugging. By efficiently exploiting program parallelism at all levels through superior execution models, efficient control, and sufficient global communication bandwidth, custom architectures exhibit high scalability to solve problems of national importance that may be unapproachable by more conventional means. Custom architectures permit a computing capability with much greater density than conventional architectures, yielding potentially dramatic reductions in power, size, and cost. Lastly, custom architectures may be the only way to achieve sufficient reliability through fault tolerant techniques for systems beyond a certain scale, which may be crucial to realizing systems in the mid to high levels of petaflops scale.

4.1.2. Challenges

Despite the promise of custom-enabled HEC architectures, there are significant challenges to realizing their potential. Foremost is the fact that—while conventional systems may exploit the economy of scale yielded by the COTS components' mass market—custom architectures, at least initially, will have only a limited market; therefore they will have fewer devices across which to mitigate development and non-recurring engineering (NRE) costs. Therefore, the benefits achieved through custom design must be able to outweigh the higher per-chip price.

The longer development time is also important because a larger part of the system needs to be designed from scratch than is the case for the COTS-based counterparts. One consequence of this is that technology refresh is less frequent for custom system architectures. There is also the challenge of user acceptance resulting from incompatibilities with standard platforms and the need to develop new software environments to address this. Difficulty in porting legacy applications, combined with the need for programmer training in the use of the new execution models and tools supported by the custom systems, can present additional barriers to both users and potential vendors. Lastly, any new system initially is unproven in the field and involves real risk to the earliest users. The introduction of any new and innovative custom system must overcome these challenges to be successful.

4.2. Summary Strategic Findings

From the wealth of facts and considerations derived from this important community forum, several key observations emerged from the consensus that should be considered in any planning of future federal programs for HEC procurement and development. The most significant ones are discussed below.

4.2.1. Advantages

Custom-enabled architectures offer significant advantages in performance and programmability compared with COTS-based systems of the same scale and deployment time for important classes of applications. A performance advantage of between 10X and 100X is expected through a combination of high-density functionality and dramatic efficiency improvements. A programmability advantage of twofold to fourfold

is possible through either the elimination or reduction of programmer responsibility for explicit resource management and performance tuning, and through advanced execution and programming models providing a reduction of sources of parallel programming errors. Significant advantages in performance to cost are expected to be yielded from the high-density packaging, low-power structures, and greater up time from intrinsic fault tolerance mechanisms.

4.2.2. Near- and Medium-Term Opportunities

Multiple technical approaches that offer significant promise for custom-enabled architectures have been identified that can be realized in the near and medium term. With necessary funding, these will accelerate computing capability and permit the United States to regain preeminence in the field. Proof of concept of more than one such innovative architecture is feasible within the next five years, and petaflops-scale computer systems can be deployed substantially before the end of this decade.

4.2.3. Strategic Partnerships

Exploiting these opportunities, so important to U.S. national security and commerce, will demand new partnerships among industry, universities, government laboratories, and mission agencies. To succeed, such alliances must be coordinated in such a way that the strengths of each institution complement the limitations of the others. Industry provides the principal skills and resources to manufacture complex computing systems, but lacks the motivation to explore high-risk concepts. University research groups devise and investigate innovative directions that could lead to future system types, but lack the resources or organization to carry them through to a useful form.

The national laboratories have the expertise of using the largest high-end computers for major applications of importance to the national welfare, but do not develop the computing engines that they use. The federal agencies have both the requirements and the resources to enable future useful systems to be invented, evaluated, and (if warranted) deployed, but have at best only limited abilities to help steer commercial technologies to niche markets such as HEC. No one side of the community can realize the opportunities of future custom architecture alone and a new class of peer-to-peer partnering relationship is necessary to restart the HEC research pipeline with new ideas, faculty, and graduate students.

4.2.4. Funding Culture

The current funding culture is incapable of enabling or catalyzing the revitalization of the HEC industry and research community. The narrow short-term specifications, limited (even single-year) time frames, inadequate budget levels, insufficient guarantees to industry as friendly customers, and conflicting objectives across agencies have dissipated the means and will of the HEC community to attempt to provide anything but incremental advances to conventional COTS-based systems, leaving future innovation to foreign suppliers. The resulting soft money mentality has largely eliminated research incentive. It has also been disruptive to national initiative compared to the Japanese, whose programs produced the Earth Simulator and are likely to deliver the first petaflops-scale computer within the next two years.

4.2.5. Innovation in System Software and Programming Environments

While it is the finding of this working group that the exploitation of custom architectures devised for the explicit purpose of scalable parallel computing is imperative to achieve the full potential of the foundation technologies, it is also clear that this alone is insufficient in meeting the goal. Both system software and programming environments must be developed to support and exploit the capabilities of the custom architectures. System software must be developed to provide the dynamic resource management anticipated by many of these architectures to improve performance efficiency and remove much of the burden from the programmers.

Programming environments must be developed that capture and expose intrinsic algorithm parallelism for greater performance, and provide high-level constructs to eliminate low-level and error-prone detail to

minimize application development time. In addition, effective software means must be provided to enable rapid porting of legacy applications and libraries to maintain continuity of the user software base. The creativity of future software and programming models must match the creativity in custom HEC architecture. The required investment in software development is likely to exceed that of the custom architecture by at least fourfold (some would estimate it at tenfold).

4.2.6. Application Requirements Characterization

Future petascale architectures, whether custom or COTS-based, will run applications of substantially larger scale and complexity than those performed on current generation massively parallel systems and clusters. In some cases, entirely new applications and/or algorithms not even attempted in the current environment may become important users of future systems. Therefore, there is little (almost none) quantitative characterization of the actual system requirements of these future systems. Against the expected sources of user demand for such systems, it has not been determined with any certainty what the resource needs will be for memory capacity, network bandwidth, task parallelism control and synchronization, I/O bandwidth, and secondary storage capacity.

It is a finding of this working group that comprehensive application studies need to be undertaken to establish quantitative system criteria sufficient to satisfy future agency demands for application performance. This is critical to the development of custom architectures, as it will establish capabilities, capacities, and correct balance for future system resources. These same results will benefit the development of COTS-based architectures as well.

Coupled equally with this is the need to convert these requirements into real metrics that allow HEC customers to adequately decide what they want and how they know that they have received it. Peak floating point operations per second (FLOPS), for example, is not enough; neither is percent efficiency, certainly not the *SPECfp* benchmark. A current DARPA-sponsored effort has begun to explore these issues, but to do it correctly will involve a broader discussion across the whole community.

4.2.7. Basic Research for the End of Moore's Law

It is expected that by 2020 the exponential growth in the density of silicon semiconductor devices, usually attributed to Moore's Law, will have reached a plateau, and that a significant reduction in the rate of performance growth due largely to silicon technology may be experienced as early as 2010 or shortly thereafter. Beyond that period, continued growth in system performance will be derived primarily through brute force scale, advances in custom computer architecture, and incorporation of exotic technologies. In this latter case, architecture advances will be required to best assimilate such novel materials and adapt computing structures to their behavioral properties.

Therefore, basic research needs to be initiated in the near future for custom architectures that will be prepared for the end of Moore's Law and the introduction of alien technologies and models. It is expected that there is the potential for significant trickle-back to silicon-based semiconductor system architecture, even before the time when such innovations in architecture will become imperative.

4.3. Technical Directions for Future Custom Architectures

Despite a period of limited funding for HEC computer architecture research, a number of paths have emerged that hold real potential; these may provide gains of one to two orders of magnitude in several critical dimensions with respect to conventional architecture and practices using current or near-term technologies. Further, it is clear that these gains will continue to prevail, at least through the end of the decade, by means of architectural and complementing system software, benefiting proportionally from enhanced semiconductor technology improvements governed by Moore's Law. This section documents key technical opportunities and potential advances that will be delivered by custom architecture research, should such work be adequately funded.

4.3.1. Fundamental Opportunities Enabled by Custom Architecture

Custom architecture uniquely is able to exploit intrinsic and fundamental opportunities implicit in available or near-term underlying technologies through innovative structures and logical relationships. Some of the most important ones are suggested here.

Function-Intensive Structures. The low spatial and power cost of VLSI floating point arithmetic and other functional units permits new structures incorporating many more such elements throughout the program execution and memory service components of future parallel system architectures. Organizations comprising 10X to 100X more functional units within a corresponding scaled HEC system are feasible in the near term, assuming logical control and execution models are devised that can effectively coordinate their operation.

Enhanced Locality. Communication is a major source of performance degradation, whether global across a system or local across a single chip. It is also a major source of power consumption. Custom architectures present the opportunity—through innovative structures to address both scales of communication, even to a significant degree in some cases—to significantly increase the ratio of computation to communication.

Exceptional Global Bandwidth. Custom HEC system architectures are distinguished from their COTS-based counterparts by interconnecting all elements of the distributed system with exceptional global bandwidth and at relatively low latency. In so doing, custom architectures can significantly reduce several sources of performance degradation typical of conventional systems, including contention for shared communication resources, delay due to transit time of required remote data, and overhead for managing the global network. Depending on the system used as a basis for comparison, improvements can easily exceed 10X and approach 100X. Such global bandwidth gains not only improve performance; they can also greatly enhance the generality of high-end systems in supporting a wide range of application/ algorithm classes, including those that are tightly coupled, are communication intensive, and involve substantial synchronization. Increased bandwidth also improves architecture scalability.

Architectures That Exploit Global Bandwidth. Bandwidth alone, although a dominant bounding condition on system capability, is insufficient to guarantee optimal global performance. In addition, custom architectures must incorporate means to support many outstanding in-flight communication requests simultaneously and, if possible, permit out-of-order delivery. This requires a combination of methods, including special lightweight mechanisms for efficient management of communication events and higher-level schema for representing and managing a high degree of computation parallelism. With high concurrency of demand and low overhead of operation, the raw exceptional capacity of custom global interconnection technology and network structures may be effectively exploited.

Efficient Mechanisms for Parallel Resource Management. A repeated requirement governing many aspects of HEC system operation is efficient mechanisms for the management of parallel resources and the coordination of concurrent tasks, especially at the fine-grain level. Fine-grain parallelism, which is crucial to scalability of future petaflops systems, can only be exploited if the mechanisms responsible for their operation and coordination are fast enough that the temporal overhead does not overwhelm the actual useful work being performed. Custom HEC architecture has the unique advantage of being able to incorporate such hardware-supported and software-invoked mechanisms employed for global parallel computation.

Advanced ISA. To facilitate the control of widely distributed and highly parallel HEC system architectures, the semantics of parallel operation needs to be reflected by the instruction set microarchitecture. This is only possible through custom design of the system and microarchitecture.

Otherwise, all responsibilities of managing concurrency of resources and tasks must be emulated through software, often requiring egregious use of synchronization variables and the overhead that entails. There are also classes of operations that—while not particularly important to general commercial computation and therefore not usually found as part of COTS microprocessor instruction sets—nonetheless can be very important to scientific/technical computing, as well as to the mission critical computations of defense-related agencies. Custom architecture may provide optimized instructions for these and other purposes that will never be available from COTS-based systems.

Execution Models That Facilitate Compiler/Programmer Application. Beyond the specifics of instructions and components, the overall operational properties of a highly scalable, efficient, and programmable parallel computing system is governed by an abstract schema for defining the relationships among the actions to be performed and the data on which they are to operate. In an actual parallel computer, such a representation formalism is manifest as an execution model that determines the emergent behavior of the system components in synergy with support of the user application. The execution model establishes the principles of control and is supported by the instructions, mechanisms, and system structure. It enables the compiler and programmer to effectively employ the capabilities of the resources comprising the system. A COTS-based system is extremely limited in the choices of execution models because they fail to provide the needed underlying functionality.

4.3.2. Examples of Innovative Custom Architectures

The working group identified several concepts for innovative custom architectures and examined their specific characteristics and advantages. Each concept incorporates structures and strategies that exploit one or more of the potential opportunities previously discussed. An incomplete set of examples of possible innovative custom architectures is presented in this section.

Spatially Direct-Mapped Architecture. An important strategy to achieve high density of functional units, low latency between successive operations, high computation to communication, and low power consumption is to enable structures of functional units and their interconnection paths to closely match the intrinsic control flow and data flow of the application kernel computation. There are several ways to do this, and the different strategies vary in their flexibility and efficiency. The “spatially direct-mapped architecture,” also referred to as “adaptive logic” or “reconfigurable logic,” comprises an array of logic, storage, and internal communication components whose interconnection may be programmed and changed rapidly, sometimes within milliseconds. The goal (and reason for the term “spatial”) is to allow us to compile not to a temporal sequence of ordered instructions, but to a spatial surface through which the data flow.

Vectors. Vector processing exploits pipelining of logic functions, communication, and memory bank access to exploit fine-grain parallelism for efficient high-performance computation. It provides a class of efficient fine-grain synchronization, the potential of overlap of communication with computation, and reduced instruction pressure. While best at exploiting dense unit stride accesses, additional mechanisms permit rapid gather scatters across access patterns that vary more widely. The vector model has been successfully exploited since the 1970s, but new implementation strategies are emerging that will extend its capability through innovative architectures.

Streaming. Streaming architectures are being proposed as an innovative strategy for providing a very-high-density logic architecture with full programmability. Wide and deep arrays of arithmetic functional units are interconnected with intermediate result data transiting through the array driven by a software/ compiler-controlled communication schedule. Very high computation to communication can be achieved for certain classes of algorithms, exhibiting high computation rate and low power.

Processor in Memory. Processor in Memory (PIM) architecture also exploits a high degree of logic density, but in a form and class of structure very different from those of vectors, streaming, and spatially direct-mapped architectures. Instead, PIM merges arithmetic logic units with memory in such a away that the logic is tightly coupled with the memory row buffers. With access to the entire row buffer, wide ALUs can be employed to perform multiple operations on different data within a single memory block at the same time. The total memory capacity of a memory chip may be partitioned into many separate units, potentially exposing greater than 100X memory bandwidth at low latency for data-intensive low/no temporal locality operation.

Special Purpose Devices. Special purpose devices (SPD) are hardwired computational structures that are optimized for a particular application kernel. They take advantage of the same mapping attributes as spatially direct-mapped (reconfigurable) architectures. But they are able to exploit very-high-speed technology and provide much greater logic density to deliver significantly greater performance per unit area and lower power per computing action. SPDs such as systolic arrays have a long history of development and are particularly useful for post sensor and streaming data applications. The world's fastest (unclassified) computer, Grape-6 [13], to be deployed within the next two years, is of this type, and it is likely that the first petaflops-scale computer will be a derivative of this architecture. An important limitation of SPDs is, as their name implies, that they are limited in the range of computations that any one of them can perform.

4.3.3. Enabling and Exploiting Global Bandwidth

Custom architectures may be distinguished from their COTS-based counterparts in part by enabling exceptional global bandwidth and its effective exploitation. Global networks for future HEC custom systems may exhibit bi-section bandwidth that is an order of magnitude greater than conventional systems, and may employ advanced technologies, including high-speed signaling for both optical and electrical channels as well as heterogeneous mixes, possibly using vertical cavity surface emitting laser (VCSEL) arrays. Optical switching and routing technologies will also be employed but it should be noted that routing and flow control are already nearing optimal capability. High-bandwidth, high-density memory devices might also facilitate fast communications. (The working group notes that the external bandwidth provided by the current generation of commodity memory devices is not anywhere near what they could be capable of using existing advanced signaling protocols, or what they already have available within the chip from the memory arrays.)

From these base technologies, advanced network structures may be created. High-radix networks organized in non-blocking, bufferless topologies will be deployable within a few years using a combination of hardware congestion control and compiler-scheduled routing strategies. A number of advances in processor architecture are key to providing a sufficient traffic stream to utilize these future generation enhanced networks for high efficiency. Within the processor control of fine-grain parallelism, architectures incorporating streams, vectors, and multithreading provide the large numbers of simultaneous in-flight access requests per processor to make good use of such enhanced global network resources. Global shared memory and low overhead message passage mechanisms make lightweight packets feasible, providing additional concurrent global network traffic. Other techniques, such as pre-fetch and pre-staging mechanisms as well as other methods of augmenting microprocessors to enhance additional requests, also contribute to the parallelism of communication and the effective exploitation of global bandwidth.

4.3.4. Enabling and Exploiting Function-Intensive Structures

Among the foremost opportunities for custom architecture are two related ones: the tremendous potential expansion of arithmetic functional units on a per die basis to increase peak floating point bandwidth by one to two orders of magnitude and, more importantly, greatly enhancing processor internal bandwidth and control locality. Spatial computation via reconfigurable logic is one such architectural method.

Streams that capture physical locality by observing temporal locality is another. New methods embodied at the microarchitecture level promise to enhance both locality and scalability of vectors. Processor-in-memory captures spatial locality via high-bandwidth local memory with low latency, and exploits high logic capability by enabling many active data/logic paths on the same chip. Chip stacking may further increase local bandwidth and logic density. General techniques of software management of deep and explicit register and memory hierarchies may lead to further exploitation of high-logic density.

4.3.5. Efficiency Via Custom Mechanisms

Efficient execution and scalability demand the ability to exploit fine-grain tasks and lightweight communication. Custom architectures provide a unique opportunity through the design of hardware mechanisms to be incorporated in the processor, memory, and communications elements. Such mechanisms can provide high-speed means of synchronization, context switching, global address translation, message generation, routing, switching, acquisition, and interpretation. Fast methods of memory management, cache handling, and security in a global parallel system can greatly reduce factors contributing to lower efficiency.

4.3.6. Execution Models

To effectively exploit the capabilities of custom architectures described in the previous sections, execution models must be devised that govern the control of the global parallel system in response to the computational demands of the user applications. A good model should expose parallelism to the compiler and system software, provide explicit performance cost models for key operations, not constrain the ability to achieve high performance, and provide an abstract logical interface for ease of programming. While no single execution model for, and supported by, custom architectures was selected, potential elements of such a future model of computation were identified based on the classes of parallel architectures being considered.

The spatially direct-mapped hardware approach suggests its own paradigm; although in the limit, it could efficiently emulate many different such models. Low overhead synchronization mechanisms open up the prospect for a rich array of parallel constructs and the potential of new memory semantics. With the prospect of PIM-enabled architectures these can be further advanced, along with additional fundamental constructs such as message-driven computation, traveling threads, and active pages. Streams and threads extend the space of lightweight efficient parallelism that both support and are supported by future execution models. Such models must distinguish between local (uniform-access) and global (non-uniform-access) memory structures and access policies.

The programming models represented by Co-Array Fortran and UPC are good first steps. However, far more sophisticated execution models will be required to fully exploit the potential of promising new custom architectures.

4.4. Open Issues

4.4.1. Programming and HEC Architecture

High-end computing is general purpose. The applications demanding HEC performance will use a wide variety of algorithms and data structures, both known and yet to be developed. Moreover, large simulations will frequently involve different sorts of models for different components, and different approaches for several time scales important to a calculation. Many applications will not match the massively parallel, data-parallel, or MPI models that are most efficiently supported by today's HEC machines. Therefore, future HEC architectures will need to support programming-in-the-large. It must be possible to put large programs together by combining components that are separately implemented. Component interfaces should be simple and independent of the mechanisms used internally. Programs for large simulations will often be so large that compiling in one step is not practical.

Some of the proposed architecture ideas are not general purpose and/or do not explain how programming-in-the-large would be supported. A large-scale, high-end system must support the simultaneous execution of multiple jobs for different users with security. This is necessary to make efficient use of the expensive equipment. Any complete architecture proposal must indicate how this will be supported. Applications people have requested both programmer/compiler direction of resource allocation and dynamic resource management at runtime. For dynamic resource management, hardware support is essential and must be included in any complete proposal. In addition, a global shared address space is essential for implementing dynamic resource management that is sufficiently general.

4.4.2. The Role of Universities

In the quest to regain U.S. leadership in high-end computing, universities provide a critical resource. Academic institutions are a major source of innovative concepts and long-term vision. They keep the research pipeline full, in part because they provide the students who are engaged in formulating and testing new ideas and developing the skills required to pursue them. Universities have demonstrated their proficiency at conducting early simulations of conceptual hardware and software systems, and they are a major facility for developing prototype tools. While it is difficult to produce leading-edge integrated circuits, universities are one of the few venues for implementing first-generation prototypes of novel concepts.

A current trend in computer science education is that students are no longer commonly exposed to massive parallelism in particular, and there is a significant decline in students of parallel computer architecture in general. In addition, there is atrophy in student interest in high-end computing. Universities provide only part of the solution and they have certain limitations. They do not do well in extending the work beyond the early research stage to the realm of robust products (there have been notable exceptions; e.g., BSD Unix). Due to the ephemeral tenure of student engagement, retaining teams is a challenge; this is aggravated by the difficulties imposed by soft money and the uncertainties of funding. This latter issue should be addressed as part of the overall strategy devised by the HECRTF.

4.5. Roadmap

Based on assumptions of sufficient and sustained funding, a general timeline of possible advances for research and development in innovative custom architecture can be projected using the concepts presented earlier as a basis for technical exploration. To this end, three epochs of five years each are considered beginning in FY05, the first fiscal year for which funding derived from this initiative may be anticipated. It is recognized that only funding for the first five-year period will be determined initially. But planning, even for this phase, requires a long-term perspective and vision to identify early basic research activities required in preparation for future conditions, such as the end of Moore's Law or the introduction of new execution models or technologies.

Three general classes of research and development are identified, including basic research, experiments and prototypes, and development toward initial deployment. At least some of the strategies under consideration can have a direct impact on systems deployed within the next five years, but will require relatively mature support software and friendly users. Additional work on these ideas will continue to refine and enhance the original approaches. The majority of architecture concepts presented will require advanced development and testing to evaluate their potential. Such experiments would result in prototypes in many cases, and yield detailed evaluation for risk reduction.

For those ideas warranting further investment, continued funding would deliver deployable architectures within the second five-year epoch. As silicon-based semiconductor technology reaches its sunset phase, truly innovative ideas must be devised and pursued that, while more risky, may ultimately and entirely supplant conventional (or even near-term advanced) approaches towards the latter half of the second

decade. In preparation for addressing HEC beyond Moore's Law, basic research will need to be undertaken even within the first five-year research phase.

It should be noted that several companies are developing custom architecture roadmaps, including IBM, Cray, SGI, SRC, as well as some smaller concerns. While the focus is on architecture, concomitant software development must be pursued in tandem with all proposed architecture research. The following sections present a sketch of a roadmap for research and development of advanced and innovative custom architecture over the next fifteen years.

4.5.1. Five Years (FY05-FY09)

Deployable. Within the first five years, specific custom architecture elements will be deployable in HEC systems delivered to government agencies prior to the end of this decade. Advances in network technology can provide a new generation of high-bandwidth interconnection links, drivers, and routers exhibiting 10X or more bandwidth, and latencies below one microsecond across very large systems. New high-bit-rate wire channels, optical fiber interconnects, and high-radix routers together can deliver critical global bandwidth gains over conventional means in real-world systems by the end of the decade. This important improvement will come with no software changes required in order to be applied to mission codes. Such work is being performed at Stanford University, among other institutions.

Symmetric multithreaded architectures will become ubiquitous within the next five years. However, there is an open question as to whether vendors will continue to emphasize single-thread performance, in lieu of supporting increased parallelism. Spatial direct-mapped architectures (i.e., adaptive logic) can be deployed as well for friendly customers, with significant advances in software and compilation strategies accomplished in this period.

Prototypes and Experiments. Several advanced architecture concepts identified previously in this report could be developed and prototyped within the first five years of a new initiative. Such experiments would permit evaluation at sufficient depth to determine which specific concepts warrant investment during the second phase to bring them to a maturity level sufficient for deployment. Examples of architecture concepts for which substantial experiments could be performed in the first five years are:

- QCDoC [13], a domain-specific architecture for high accuracy QCD calculations at Columbia University,
- Merrimac Streaming architecture for high computation to communication processing at Stanford University,
- The Berkeley Emulation Engine [6], another domain-specific architecture for immersed boundary method codes at UC Berkeley,
- PIM-lite and MIND processor-in-memory (PIM) architectures for general-purpose, data-intensive (low temporal locality) computations at the University of Notre Dame and Caltech; and
- Ultra-low-latency optical networks using fiber optics, at Columbia University.

Basic Research and Exploratory Studies. Beyond the continued current trends of silicon-based semiconductor technology dictated by Moore's Law, innovations in device structures and technologies and radical changes in architecture and execution models will require that fundamental basic research be initiated within the first five years of the new program. The early exploratory studies will develop inchoate concepts and push the edge of the envelope to provide the mission agencies with alternatives to avoid limits on capabilities in the early part of the second decade. Such research could include new computational models, nanotechnology, quantum dots, cellular automata, amorphous computing, and continuum computer architecture.

The purpose of these studies would be to develop the basic concepts and relationships among these new technologies, structures, execution models, programming models, and application algorithms capable of

exploiting billion-way parallelism towards exaflops-scale computation. New architectures that are resilient, fault tolerant, and with decentralized control (e.g. distributed agents) would yield robust systems at scales where single point failure modes would limit sustained uptimes to seconds. New approaches to compilers and runtime systems as well as scalable I/O and operating systems would be undertaken, and would be strongly influenced by work on novel programming models.

4.5.2. Ten Years (FY10-FY14)

The second five-year epoch of a new funding initiative in custom computer architecture could prove to be an explosive renaissance in system design, capability, robustness, and usability. All prior prototypes and experiments that had demonstrated viability during the first five-year phase of the program could be deployed at mission agency sites. Such systems would provide sustainable performance for general applications in the 10 to 100 petaflops performance regime and exhibit competitive recurring cost compared with conventional techniques. They would also deliver far superior operational attributes for at least many important agency applications, assuming they were properly funded. Virtually all of the prior technology opportunities developed in the first phase will be deployable by the second phase in real-world HEC systems. But initial adoption of such systems will be limited by drastic changes required in execution and programming models, although methods for transferring legacy codes to such radical systems will be a continued area of important research. Infrastructure will have to be established between academia and industry to encourage and enable the transfer of research results and their incorporation in deployed hardware and software systems.

Innovative concepts, having gestated during the first five-year phase of the project, will be down-selected based on probable risks and potential rewards, with the most promising opportunities carried forward into the experimental phase of research with possible prototypes being implemented and tested as appropriate. For example, a continuum computer architecture prototype implemented with quantum dots fabricated using nanoscale technology might be prototyped to establish the feasibility of manufacture and application. It is this set of experimental systems that will set the stage for the post-Moore's Law era to sustain the growth of HEC system capability, in spite of the flat-lining of conventional semiconductor technology. During this period, new concepts can be anticipated from academic research groups, and these will provide the basis for new basic research projects.

4.5.3. Fifteen Years (FY15-FY19)

With silicon scaling at sunset, systems developed and deployed during the latter part of the second decade will exploit revolutionary techniques in circuits, packaging, architecture, and software strategies. These truly revolutionary custom architectures will mesh with the end of the silicon roadmap and new non-silicon technologies that will have been proven during the previous phases of the program. As these exotic systems are prototyped and deployed, alien in form and function but capable of near-exaflops performance, entirely new software environments for resource management and programming will have been devised and will be developed during this period. Ironically, these systems delivering one hundred thousand times the performance of current HEC systems may be smaller, consume less power, and take up much less space than even today's terascale systems. We cannot know this part of the roadmap, but we can prepare for it and enable its extraordinary impact through basic research in the first phase.

Regarding the basic research to be conducted in this end phase, we cannot even hazard a guess. For if we were able to speculate at all, we would be defining part of the research agenda for the next five years, not one of more than a decade from now. But we can be certain that with sufficient and sustained funding, the HEC community in the United States can provide the new ideas, technologies, and applications that will continue to drive this nation's competitive position for defense and commerce as this century enters its adolescence.

4.6. Summary and Conclusions

Custom-enabled HEC architecture provides a vital alternative to conventional COTS-based system design by enabling the exploitation of potential advantages intrinsic to available and near-term technologies, but demanding innovative hardware structures and software management models and methods. In many important metrics, custom architectures may deliver between 10X and 100X advantage over conventional COTS-based systems employing equivalent semiconductor technology including peak and sustained performance, performance to cost, power, size, and reliability. Custom architectures may efficiently support advanced execution and programming models that will both deliver superior sustained performance and greatly facilitate programmability, thus enabling systems of exceptional productivity for applications driven by federal agency missions.

It is imperative that research in advanced, custom scalable HEC architecture be sponsored at an accelerated and continuous level to regain U.S. leadership in the field of HEC architecture, and provide the tools to secure dominance in this strategically critical technology for national security and commerce. To a large extent, the students we train in the first epoch will be the ones doing this final epoch work, and it is crucial that we give them the mindset, tools, and funding to be able to set a truly innovative and aggressive research plan fifteen years from now.

5. RUNTIME AND OPERATING SYSTEMS

Rick Stevens, Chair
Argonne National Laboratory

Ron Brightwell, Vice Chair
Sandia National Laboratories

The working group considered the principal functional requirements of operating systems and runtime systems for high-end computing systems likely to be available by the end of the decade. The group also explored the research needed to address these functional requirements. Lastly, the group discussed the role of open source software and the need for testbeds to enable broad community participation in research and development and to support outreach and education.

To focus the discussion, the working group made the following general assumptions. First, we assumed that future systems would be substantially larger than current systems. Hence, the scalability target for analysis considered systems several orders of magnitude larger than today's largest systems (i.e., one hundred thousand to one million nodes). We also explicitly targeted both COTS-based systems and systems made from custom logic, including commercial systems. At each point, we considered current technologies, the limitations of current approaches, leverage that might be obtained from open source software technologies, and new ideas needed for breakthroughs. Where possible, we indicate those recommendations that could be pursued in the near term and those that are longer term due to either technological unknowns or problem difficulties.

The working group considered a broad range of topics relating to runtime and operating systems for high-end computing systems, including operating system and runtime Application Programming Interfaces (APIs), high-performance hardware abstractions, scalable resource management, file systems and data management, parallel I/O and external networks, fault management, configuration management, operating system portability and development productivity, programming model support, security, operating system and systems software development testbeds, and the role of open source software.

5.1. Recurring Themes

The working group noted the limitations of UNIX (Linux and other UNIX variants) for HEC. Unix has been the technical world's operating system of choice for nearly thirty years. While it has served the community well, its very design point and set of assumptions are increasingly at odds with the needs of high-end computing. Today, the operating system and runtime research community is almost entirely focused on delivering capability via commodity-leveraging clusters—this may not be the proper balance for the future.

We believe the operating system and runtime service models will soon merge. We also believe that increasing performance feedback for dynamic adaptation, including increasing coupling among operating system, runtime, and applications, is an important trend. Increasing the transparency (i.e., exposing and making visible) of those aspects of the software and hardware that impact performance is increasingly important if we are to manage very-large-scale systems effectively.

We also believe many groups will opt for a minimalist approach to operating system and runtime services, whereas others will need operating system support that is more fully featured. Future systems should be capable of supporting both models. There may also be opportunity for improving operating system and runtime performance via hardware support for certain features and functions (e.g., fault

management and resource monitoring). We also believe operating system and runtime research should be more closely coupled to hardware research in the future.

Lastly, it was very clear that the current lack of large-scale testbeds is limiting operating system and runtime research for the HEC community. Such testbeds can provide the broad research community with access to flexible testing environments and scalability research platforms. These testbeds should be available to university groups, national laboratories, and commercial software developers.

5.2. Operating System Interfaces

It seems clear that the POSIX APIs used on most operating systems will be inadequate to support future large-scale HEC systems. In particular, the overhead of some POSIX operations varies widely across systems, leading to unexpected performance problems when moving applications from one system to another. Some POSIX operations also require a heavyweight operating system on each node, which is inappropriate for systems composed of large numbers of lightweight processing elements. The need to run current applications on future HEC systems requires that these systems continue to support a POSIX compatibility mode, but this need not be the primary API.

An ideal operating system API would have performance transparency so that the cost of every operation is visible to the programmer, compiler, or code generation system. Such an API should also be sufficiently modular to support HEC systems that do not run a full operating system on every node. Future applications will still require common APIs across operating systems for application portability; vendor-specific APIs are not acceptable. Because there has been little work on common, high-performance, non-POSIX APIs, we recommend a research initiative to develop such APIs. In the longer term, a new operating system API must be standardized and implemented by HEC vendors.

5.3. Hardware Abstractions

Historically, one operating system role has been abstracting the underlying hardware, providing a virtual machine that either contained additional features or hid hardware idiosyncrasies. The main drawback of virtualization is that it hides key features of the hardware that the application may be able to exploit for improved performance.

As an example, consider two programming models, one based on shared memory using threads and the other based on MPI. In the threaded model, the operating system can hide the number of actual processors from the application by multiplexing threads on the hardware. This simplifies the programming model and enables support for dynamic allocation of processing resources as the number of threads changes. The drawback is that multiplexing threads on processors can lead to high overhead for context switching and cache thrashing. In MPI programming, it is common for the number of MPI processes to equal the number of processors on which the program is running. However, when debugging, it is common to place a large number of processes on a small number of processors. The user must make an operating system call to determine the current execution mode.

Lastly, as hardware becomes more varied, the opportunities for abstraction increase. For instance, it is not obvious that an operating system should hide the number of PIM processors or the details of an FPGA extension. If the hardware changes or is unavailable, what behavior is appropriate? Additional research is needed to identify the appropriate virtualization and abstraction models. These models must elide unnecessary detail while also providing performance transparency—the ability to optimize for hardware details where needed.

5.4. Scalable Resource Management

Operating system services for resource management must adapt to the usage model of high-performance applications. In particular, cluster-level resource management should schedule cluster resources in an

optimal way for parallel application performance. The cluster resource manager should map application processes and threads onto the system in a way that optimizes message traffic, given knowledge of application messaging dynamics and network topology. Similarly, node-level resource management should provide a minimal set of operating system services that maximizes the resources available for application use.

As an example of intelligent resource management, processes that generate large volumes of message traffic should be placed nearby on the network to minimize message latency. More generally, a cluster resource manager should schedule network and I/O resources (to disk or archival storage) to avoid resource contention, provide predictable performance, and avoid ailing nodes. Lastly, the time needed to allocate resources and launch a large parallel job should scale logarithmically with the number of processes.

Typically, scientific applications do not share node resources, and they are sized to utilize all of the node's available computing resources (i.e., memory, processors, and network interfaces). Thus, sophisticated operating system services designed for commercial use (e.g., such as virtual memory management and time-sharing) are not appropriate for high-end computing. Rather, it should be possible to manage a node's available memory in application space. NUMA memory systems should minimize the distance between memory placement and the processor on which the process/thread is running. Once memory is allocated, it should not move in the non-time-shared environment of node-level HPC.

Operating systems for high-end computing must also manage shared resources. Examples include files (long-lived data) and the communication network in systems with multiple, concurrently executing applications. Where possible, this protection should be implemented in hardware. When hardware protection is not provided, the needed protection must be implemented in software. Currently, high-end systems do not provide adequate hardware to control access to the communication fabric; these systems require a software layer to provide the necessary access control. In contrast, the communication network of the planned IBM Blue Gene/L system can be partitioned to reflect the node allocation when an application is launched.

Lastly, applications should have the flexibility to manage resources on their behalf when such access does not compromise inter-application security. Otherwise, performance-sensitive applications incur unnecessary complexity and overhead to implement their own version of resource management. In these cases, performance is substantially worse than if the operating system implemented the right policy.

5.5. Data Management and File Systems

We believe legacy, POSIX I/O interfaces are incompatible with the full range of hardware architecture choices contemplated. The ordering semantics for multiple, simultaneous writers are particularly onerous on a distributed memory system that lacks implicit memory synchronization. Additionally, the interface does not fully support the needs for parallel support along the I/O path. For instance, implementations leverage non-standard extensions and overload to communicate striping needs and strided access.

Any file system suitable for use on high-end systems must be extremely scalable. The distributed, cooperative problem-solving approaches assumed by clusters and multi-program, parallel platforms have ignored the potential processing capability within the I/O system. An alternative, appropriate operating system API should be developed for high-end computing systems. However, such an alternative should not deviate from POSIX except where POSIX functionality limits performance or programmability.

Data management systems should be developed that leverage enterprise-wide authentication schemes and provide interoperable authorization mechanisms. They should recognize the differences in capacity and performance required by high-end systems and workstations, while functioning appropriately in the

presence of both. In general, they should promote sharing via primitives to access data across geographically distributed administrative domains.

Lastly, the concept of a distributed approach to problem-solving could be extended to include the hardware closest to the storage. Intelligent storage services such as active disks could yield significant throughput enhancements by selective filtering, deep pipelines, providing translations, or other small pre- and post-processing duties. Research is also needed to explore appropriate methods for decomposing metadata services to enable scalability.

5.6. Parallel and Network I/O

Some classes of future HEC systems will have hundreds of thousands or millions of processors. High-speed, specialized interconnect fabrics will provide communications among processors or groups of processors. In such systems, each processor may not have an external network or I/O channel interface. Instead, many processors, possibly thousands, will share a common external network or I/O channel interface. Operating systems and/or runtime systems will be required to share, schedule, and control these resources. There are several possible ways to provide this service, including gateway nodes, proxies, routing, direct protocol conversion, protocol layering, and multi-protocol switching technology.

Because these future high-end systems may have thousands or even millions of external I/O or network channels and some of the I/O resources may be geographically distributed, Grid and network technologies may be needed for remote resource management, including security as well as scalable and dynamic parallel use of external network and I/O interfaces.

Hence, research is needed on mechanisms that can provide suitable shared, external I/O and general network interfaces for HEC systems. Because many HEC sites currently have this problem, some coordination of these ongoing research activities would be helpful to collect a body of useful information to determine if there are general answers. Over the longer term, perhaps general research into aligning internal mesh and external I/O and network interface protocols is in order. Additionally, research into the proper way for HEC systems to interface with the Grid services is needed, especially research to validate the appropriateness of solutions at scale. Lastly, scheduling bandwidth, quality of service, and fairness are also issues to consider.

5.7. Fault Management

Handling faults is critical to the future of high-end computing, and they can occur in many different system components: memory, interconnects, disks, and nodes. To manage such faults, integrated solutions involving detection/prediction recovery are needed. Some solutions exist today for handling memory, interconnect and disk faults. Moreover, applications-level mechanisms (e.g., check-pointing and recovery) are used to handle node faults. However, as system sizes increase to 100,000 nodes and beyond, novel scalable and high-performance solutions are needed to manage faults and maintain both application and system operation.

The working group recommends several, concurrent research tracks for efficient and effective fault management in large-scale, high-end systems. First, efficient schemes, including operating system mechanisms and architectural solutions, are needed for fault detection and prediction, diagnosis, and recovery. Second, novel schemes such as replication and ubiquitous virtualization must be included in the runtime system to handle faults gracefully, with little impact on performance. Lastly, integrated solutions with configurable management schemes are needed to reduce the cost of detection and recovery, while also minimizing performance degradation.

5.8. Configuration Management

Current configuration management tools for high-end systems are inadequate in several ways. First, the tools do not scale well to systems with hundreds or thousands of nodes, and there is little prospect they will perform well on systems with tens of thousands or millions of nodes. In the event of failures during a system update, an inoperable system may result or the state of the nodes of the system may be unknown. The failure of an update may require extensive individual installation and/or repair of software for each node that was incorrectly configured. These shortcomings should be addressed if large-scale systems are to be manageable in a cost-effective, productive way.

We recommend that system software release and configuration management tools be able to operate effectively at both moderate and very large scales. We further recommend that reliability be a major concern during the development of systems release management software. In the event of a failure of all or some of the nodes during a systems software upgrade, the cluster should be in an operational state. It should be possible for the systems manager to determine which nodes have been updated and which have not; and it should be possible to restart the upgrade operation from where the operation was interrupted.

Nodes that were unavailable during the upgrade should automatically update themselves when they are brought back online. Furthermore, it should be simple for a manager of a very-large-scale system to revert to a previous configuration or move forward to a test configuration of the systems software on all or any subset of the nodes.

In the short term, reliability should be favored over speed and scalability. It is important to develop capabilities such as journaling or multiphase commits to ensure that an operable system results from an upgrade operation. It should be possible for all nodes to maintain an inventory of their state and to ensure that they are up to date at boot-time or when they are brought into service. After these basic operations are developed, scalability of these operations needs to follow shortly thereafter. In the longer term, standardized interfaces for configuration management should be developed to allow for different packages that comprise a “system” to be integrated and managed with the same tool set. We may need to deviate from the file-based Unix approach.

5.9. Operating System Portability

As has been noted repeatedly, the next generation of high-end systems will be composed of large numbers of components—compute nodes, storage nodes, administrative nodes, switches, monitors, and management devices—each with its own local processor. It is highly desirable that these components have common features implemented using a shared code base. The primary distinctions among the components will be their degree of customization with respect to their host applications, and the set of devices controlled by their local processor. Components may also be customized with special-purpose hardware, such as FPGAs.

To the degree that the component control framework and shared features (such as RAS, firmware loading and network interfaces) can be made common, high-end systems will benefit from reduced costs and increased productivity. This core operating system will then provide the basis for the runtime platform for each component. Shared code is also likely to be more reliable as well.

One of the Linux operating system’s strengths is the wide variety of hardware device-driver code available. One obstacle in the development of new or improved operating system software is the need to modify or recreate a variety of device drivers. Device drivers are non-portable for several reasons. First, the context in which they are embedded (e.g., a network protocol stack such as Bluetooth) is insufficiently parameterized to allow the substitution of alternate drivers. One example of this was the proliferation of entire TCP/IP stacks on Windows systems before the movement of the TCP/IP stack into the operating system. Second, the context in which the driver executes differs across operating systems, and varies over

time within an operating system. For example, the memory management functions within the Linux kernel changed in the midst of version 2.4; drivers that worked in 2.4.3 no longer worked in 2.4.9.

One can view the role of the device driver within a protocol stack (or any use of the driver) as the provisioning of an actual strategy (the driver implemented as a module) into a parameterized module (the network stack). This requires a well-defined interface with well-designed architecture, along with the protocols for how the modules communicate and interact. The adoption of such a "strategy pattern" is useful at many levels within the operating system software for HEC. For example, consider a job-scheduling system that is parameterized with a module providing specialized node allocation based on known network characteristics. The scheduler itself could then be independent of the actual network topologies.

Modularity is not a new issue, either in the design of systems or programming languages. What is needed is to find and use efficient and effective techniques for developing and documenting parameterized operating system modules, while minimizing the performance overhead incurred. This approach does not require object-oriented inheritance, but only a distinction between interface and implementation, along with the ability to bind alternative implementations.

We recommend a review of current driver implementations to abstract both execution context requirements and to derive interface requirements. We must also develop processes to encourage the adoption of clean, well-defined, and long-lived models for device interfaces. Concomitantly, we must support and encourage the development of a common runtime execution platform that can be used as a basis for all processor-equipped devices within a large system, supporting common services such as RAS, monitoring, or configuration management.

We must also encourage the definition and adoption of parameterized modules throughout the design of all operating and runtime systems. Moreover, we must invest in the use of modular programming languages for implementing operating systems, including the development and deployment of tools needed to improve the runtime performance of parameterized modules. For example, a binding tool can be used to eliminate performance overhead when the parameterization is set during system link time.

5.10. Operating System Security

Several definitions are in order before discussing operating system security. Multilevel security (MLS) processes information with different classifications and categories that simultaneously permit access by users with different security clearances and deny access to users who lack authorization. Second, authentication is the process whereby an entity proves its identity to another entity. Lastly, authorization is the set of access rights granted to a user, program, or process. It is also the operational act of allowing subjects to access a resource after determining that they hold the required set of privilege attributes.

Almost all operating systems currently in use rely on a security model based on the original Unix security model—resources are accessed via processes. Resources are represented as files (devices even look like a file), and they have an attribute consisting of a user name and group name. Each resource has a permission mask, which defines access permissions (read, write, execute, and others) for the user name, group name, and all others (world). Processes are authenticated via an initial process (usually login), which sets the user and group identifier of the process as well as a set of other groups. From that point, the process attributes allow access from the process to any resource that matches those attributes.

In Unix, processes cannot re-authenticate and change authorization; the only mechanism for a change in authorization is the setuid system call, a blunt tool indeed. Processes cannot present different authentication data to different resources to gain different authorizations. Resources are not active in

Unix; they are passive, and hence cannot engage in authentication or authorization activity with a client. Moreover, resources cannot possess attributes outside the simple user/group/world model.

The file system space of a Unix system can be described as a "space of names"—a name space. Unix processes augment the name space via the Unix *mount* system call. When a process modifies the name space of a machine, the modification is global—all processes on the system will see the modification. In programming language parlance, mounts are not free of side effects—they modify the global state of the system, not the local state of the process. Unintended sharing can be the result, violating the goal of confinement.

Lastly, the Unix user-naming mechanism is outmoded. Users are named, and permissions determined, by an integer number. There is obviously no possibility of gaining agreement on user name to number mappings on a global scale; it is hard enough in a single organization. In Grid environments, this integer user ID (UID) presents problems that have been widely discussed. The designation of a user by an integer also causes problems in accessing file servers such as NFS, since the UIDs must be kept consistent across the client/server boundary, and such consistency is again unachievable on the Grid. The entire file system layer of Linux revolves around the UID mechanism and is not easily changed to some other mechanism. Modifying Linux to use names, not UIDs, would require an extensive rewrite of the kernel, GNU library, and GNU tools, as well as almost all extant network file systems.

Finally, Unix operations rely on root as a privileged user. For the root user, no operation is off limits or out of bounds. Privilege checks are bypassed for the root user. The root user can attain the privileges, via setuid, of any other user. Most Unix exploits revolve around spoofing programs running as root into doing something that violates the security on the system. The existence of the root user will, inevitably, reduce the security of the system. Removing root is impossible because so many basic Unix mechanisms rely on the existence of root.

The lack of sophisticated permissions in Unix limits the ability to implement strong security and multi-level security. Because resources are passive not active, they cannot engage in more sophisticated authentication transactions. Because resources have simple permissions, the increased sophistication required for MLS and more fine-grained access controls is not possible. Because name spaces are global, unintentional leakage of information ("side effects") is possible. The limitations of integer UIDs are pervasive, although most Unix-like operating systems use them for high-end systems. The structure of Unix is difficult and in some cases impossible to modify, since many of these properties are structural.

Existing Linux extension efforts do address some of these problems. Linux will soon have Access Control Lists (ACLs), which will enhance the protections provided by the file system. Since Linux version 2.4.19, Plan 9-style private name spaces have been implemented on Linux in a limited way. The V9FS project (http://v9fs.sourceforge.net) builds on this capability by providing 9p2000 clients and servers, allowing processes on Linux to augment their name spaces with strong authentication tools. This separates authentication and authorization from the protocol, moving these low-performance bottlenecks out of the main file I/O code. In the end, however, the Unix security model is not extensible. If nothing else, the use of integer UIDs and the existence of the root user would be sufficient to require replacement of the Unix model.

What new models might be used in future operating systems? Several possibilities are identified, such as: making resources active instead of passive; allowing processes to reauthenticate as needed for different resources; allowing processes to present different authentication, and gain different authorization, for different resources; providing a richer and more fine-grained access control model; supporting private name spaces; and better support for Grid requirements by eliminating (e.g., integer UIDs and the privileged, or root, user).

There are two operating systems, Eros and Plan 9, that provide a proof-of-concept that such ideas can be achieved. Eros [25] (http://www.eros-os.org/) is a new, capability-based operating system. In Eros, access to resources is controlled by capabilities attached to the resource. Resources are passive. We cannot yet tell whether integer UIDs are completely gone, but integer UIDs are inconsistent with the use of capabilities, so we are guessing that they are. Root has no meaning in a capability-based system; we have not yet audited the Eros code, but we believe there is no analogue to the Unix root user.

Plan 9 (http://plan9.bell-labs.com) provides a very different system model than either Unix or Eros. In Plan 9, resources are active. Processes cannot change their identity once it is established, but they can present different authentication, as needed, for gaining access to resources. Plan 9 invented the concept of private name spaces. Plan 9 separates authentication and authorization from operations on the resource, which is desirable for HEC. Plan 9 does not support fine-grained access to the resources; it has not been missed due to the way private name spaces work. Plan 9 has no root user.

It is clear that the Unix process and resource model has limitations that affect HEC environments, and it is not sustainable for the long term. Ongoing changes to Linux, via the provision of ACLs, private name spaces, and V9FS, show that in the short term some of the problems may be ameliorated. It is also wise not to underestimate the willingness of the Linux kernel team to make far-reaching changes to the Linux, as the 2.5 and 2.6 kernels show. Nevertheless, there is no clear solution to some of the fundamental problems, such as the existence of the root user. The two example systems presented show that very different models are possible and viable (Plan 9 is used widely in internal product development at Lucent). Research into new security models could lead to resolutions to these problems.

Hence, we believe that new research is needed to explore alternate security models. This should explicitly include support for operating system models different from Unix.

5.11. Programming Model Support

The predominant parallel programming model currently in use is based on message passing using MPI, a fifteen-year-old technology. MPI relies on a library of routines to manage low-level details of parallel execution, and it requires the developer to partition an application into many independent cooperating processes to exploit process-level parallelism. This imposes a considerable intellectual burden on an application developer. Although MPI programs are portable across many execution platforms, the development costs are high.

Two alternative programming systems, Co-Array FORTRAN (CAF) [22] and Unified Parallel C (UPC) [9], are worthy of concerted experimentation. These promise ease of use and reuse, as well as gains in programmer productivity and the possibility of high performance. Other possible advantages include increased tolerance for processor latency, reduced overhead through single-sided communication, and shared name spaces.

Additionally, application development productivity can be enhanced through better compilers, debuggers, and performance analyzers. As future processor architectures introduce greater internal concurrency and deeper memory hierarchies, compilers will be required to better insulate the programmer from such details. Similarly, debugging and performance analysis can enhance productivity through automated or semi-automated tools. The runtime environment need to be enhanced by new or improved tools described above. The programming environment offering CAF and UPC may need additional capabilities from the operating system for sophisticated management of memory.

6. PROGRAMMING ENVIRONMENTS AND TOOLS

Dennis Gannon, Chair
Indiana University

Richard Hirsh, Vice Chair
National Science Foundation

The charter of this group was to address programming environments for legacy codes and alternative programming models to maintain the continuity of current practices, while also enabling advances in software development, debugging, performance tuning, maintenance, interoperability, and robustness. Our goal was to identify key strategies and initiatives required to improve time to solution and ensure the viability and sustainability of HEC systems by the end of the decade.

The working group considered several possible approaches to improving future programming environments. These ideas ranged from innovations that support incremental evolution of existing programming languages and tools, consistent with portability of legacy codes, to innovative programming models that have the potential to dramatically advance user productivity and system efficiency/ performance.

6.1. Key Observations

The key findings of the group can be summarized as follows. The most pressing scientific challenges of our time will require application solutions that are multidisciplinary and multi-scale. The complexity of these systems will require an interdisciplinary team of scientists and software specialists to design, manage, and maintain them. Accomplishing this task will require a dramatic increase in investment to improve the quality, availability, and usability of the software tools that are used throughout the lifecycle of the application, which will span many generations of HEC platform architectures.

The strategy for accomplishing these goals is not complex, but it requires a change in attitude about software funding for HEC. Software is a major cost component of all modern, complex technologies, but the tradition in HEC system procurement is to assume that the software is free. Mission critical and basic research software for HEC is not provided by industry because the market is so small and the customers are not willing to pay for it. We need federally funded management and coordination of the development of high-end software tools for high-end systems.

Funding is needed for basic research and software prototypes, and for the technology transfer required to move those prototypes that are successful into real production-quality software. We need better ways for interdisciplinary teams to collaborate and to integrate well-tested software components into working systems. It is urgent that we invest in building interoperable libraries and software component and application frameworks that simplify the development of these complex HEC applications. These technologies show great promise in revolutionizing HEC programming methodology to improve time to solution. It is also essential that we invest in new, high-level programming models for HEC software developers that will improve productivity, and create new research programs that explore the hardware/software boundary to improve HEC application performance.

Structural changes are needed in the way funding is awarded to support sustained engineering. We need a software capitalization program that resembles the private sector in its understanding of the software life cycle. One approach to coordinating federal effort in this area would be to establish an institute for HEC advanced software development and support, which could be a cooperative effort among industry, laboratories, and universities.

Lastly, a new approach to HEC education is also needed, including a national curriculum for high-performance computing. We need continuing education that will enable us to build interdisciplinary science research. To enable an improved educational agenda, we need a national HEC testbed for education and research that will provide both students and researchers unfettered access to the next-generation HEC systems.

6.2. The State of the Art and an Evolutionary Path Forward

The term "legacy software" refers to computer programs that embody the state of the art of our scientific understanding. Like our scientific understanding, legacy codes are seldom static; they are constantly being modified to reflect the evolution of our understanding. Consequently, it is our duty to maintain them on each new HEC platform. Unfortunately, the core modules of legacy code are written in old programming languages, and they are often designed using outdated software construction principles. To sustain them, new modules are carefully grafted on or parts of them are manually restructured to make them work on new architectures. New approaches are needed to help manage the "life cycle" of these evolving applications.

As our scientific understanding grows, new legacy programs continue to be created. In addition, algorithmic advances have been required to solve more complex multi-scale problems. These new programs use a variety of programming tools, including FORTRAN (66 through 95), C/C++, and special interpreted scripting languages like Python and MATLAB. Often, new applications require a blend of all of these and, to get performance on parallel machines, it is necessary to use libraries like the Message Passing Interface (MPI) and explicit threading, or language extensions like OpenMP and High-Performance FORTRAN (HPF). Currently, MPI is the most common high-performance programming tool. For many computations, MPI leads to excellent speedup, but in other cases, it requires casting problems and algorithms into an unnatural form where performance scales poorly.

Many researchers once hoped that compiler-based, automatic parallelization would solve the problem of scaling sequential software to massively parallel computers. While we have made great strides, a complete solution is not in sight. However, automatic parallelization has proven to be an essential tool for optimizing procedure bodies, and for generating the low-level parallel code that is executed on modern processors. One way to make it easier for the compiler to generate parallel code is to use programming language extensions that allow the applications developer to express "top-level" parallelism that can be used by the compiler. These new language extensions can be considered an important evolutionary path forward. They include Co-Array FORTRAN, UPC, Adaptive MPI, and specialized C++ template libraries.

6.2.1. Software Productivity

Clearly, progress is very slow in evolving high-end software practices to new languages and programming models. The rest of the software industry moves much faster in the adoption of new software development paradigms. Why is high-end system software development different? Scientists and engineers continue to use older approaches because they are still perceived as the shortest path to the goal: a running code. Many of the tools are well suited to the traditional model of one-programmer-one-program. However, our high-end system applications are rapidly evolving to multi-language, multidisciplinary, multi-paradigm software systems that are built by distributed, collaborating teams.

The complexity of these applications will soon rival that of large projects at companies like Oracle, SUN, or Microsoft, where software engineering standards, practices, and tools are major corporate investments. High-end computing has been largely deprived of participation in the revolution in software tools. When tools have been available, centers cannot afford to buy them. When they are available, they are not

available on all the requisite HEC platforms. For example, industrial-strength "build, configure and testing" tools are not available for most HEC applications/languages.

Near-Term Solutions. A major initiative is required to improve the software design, debugging, testing, and maintenance environment for HEC application software systems. We need portable software maintenance tools across HEC platforms. We also need a rapid evolution of all language-processing tools, and we need complete interoperability of all software life-cycle tools.

Performance analysis should be part of every step of the life cycle of a parallel program. For example, feedback from program execution can drive automatic analysis and optimization of applications. Developing extensible standards for compiler intermediate forms and object-file formats can facilitate interoperability. As an example, Microsoft provides excellent interoperability for all of its languages and tools. The HEC world is, by its nature, far more heterogeneous than the monolithic world of Microsoft, making our problem much more difficult. However, our long-range goal should be complete "roundtrip engineering" of HEC software (i.e., the ability to take a specification of a computation and automatically convert it to executable form and then back again for design changes to improve either the science or the performance).

Evolution of HEC Software Libraries. The increasing complexity of scientific software (multidisciplinary and multi-paradigm) has other side effects. Libraries offer an essential way to encapsulate algorithmic complexity, but parallel libraries are often difficult to compose because of low-level resource conflicts. Older libraries often require low-level interfaces that do not exchange more complex and interesting data structures. These problems have led researchers in some important new directions.

Software component technology and domain-specific application frameworks provide a way to put multi-paradigm parallel programming within reach. These tools provide an approach to factoring legacy into reusable components that can be flexibly composed. The framework handles resource management while components encapsulate algorithmic functionality. These systems provide for both interface polymorphism and system evolvability. They also allow us to abstract the hardware/software boundary so specialized hardware can replace a software component without changing the rest of the application. In general, these component systems approaches allow better language independence/interoperability. Testing and validating applications are made easier because we can test individual components separately before they are composed into a larger system. We can also ensure components can be trusted, and, if the framework that hosts the components is well designed, we have more trust in the entire application.

Interoperable component technology made a marketplace of integrated circuits possible; software component technology may also create a market for HEC application components. While clearly still a long way off, this would be an exciting outcome. It may be faster to build a reliable application from reusable components, but will it have performance scalability? Initial results with software components based on parallel computing indicate that the answer is "Yes." However, more research is needed.

6.3. Revolutionary Approaches

A long-range program of research is needed to explore new programming models for HEC systems. The scientific programming languages of the future should allow the scientist to think about science rather than the intricacies of parallel programming. One should be able to express concurrency as the natural parallelism in the problem and not as an artifact of a software model.

For HEC architectures, the challenge is exploiting locality and hiding communication latencies. However, in large-scale simulations, locality is natural to the problem, but it is often non-regular or dynamically changing. For high-end systems with limited memory hierarchy management and latency-

hiding support, software must manage latency. Unfortunately, we are very far from building tools that are sufficient for this task.

One solution is to design languages from first principles to support the appropriate abstractions for scalable parallel scientific codes (e.g., ZPL [5]). In general, we must consider all approaches that promote automatic resource management. For example, we can build extensible compilers that allow us to integrate user-domain abstractions into the compilation process. Another approach is to telescope languages [7] that allow us to build application-level languages that compile to a sequence of lower-level languages, each of which can be optimized for a particular class of parallelism and resource management.

Many concepts that have gained currency in computer science can be applied to high-end computing to solve both the problems of performance scaling and HEC software life-cycle management. For example, generic programming teaches us how to separate data structures and algorithms when building software. This allows us to reuse the pattern of the algorithm while we replace data structure components. In some cases, it may be possible to automatically generate program components, given a specification of both the algorithm and the data structure. This also allows us to publish and discover algorithms as reusable software components that can be separately coupled with data structure components.

Another example of a potentially rewarding approach to high-end system software is "programming by contract." Here, the programmer specifies the requirements of the generated code and the compiler works to either attain that goal or report to the programmer the trade-offs needed to achieve the goal. For example, the programmer may have a certain performance level that is required or an absolute requirement on correctness/repeatability. Alternatively, there may be a requirement for robustness that may necessitate the introduction of a persistence model into the program.

6.3.1. Research on the Hardware/Software Boundary

It remains difficult to achieve high application efficiency on large-scale cluster architectures, and only a handful of applications scale to several thousands of processors without careful hand-tuning. Much more research is needed to explore the detailed interactions between the hardware and software on modern HEC systems. To accomplish this, we need better instrumentation on all parts of the hardware. This includes performance counters for the computation units and the memory hierarchy.

We must also explore programming language-type systems that better reflect the properties of HEC architectures. As the operating systems working group noted, we need an open, bi-directional API between the hardware and software. For example, one may wish to change the memory consistency model depending on the application. In the future, we may have more processors based on reconfigurable hardware so an even greater burden/opportunity is presented to the software.

Predictability for scheduling, fine-grained timing, and memory mapping is essential for scalable optimization. As HEC systems grow larger, fault tolerance also becomes more and more important. Software need to be more aware of the dynamic nature of its execution environment and have the ability to adapt to change. We must develop programming models that better support non-determinism (including results that are desirable, but boundedly incorrect).

We need a better understanding of how the complex memory hierarchy of modern systems interacts with our software and algorithms. There are limits on what is possible with legacy codes that have poor memory locality. Software needs better mechanisms to map data structures to memory hierarchies. Possible new solutions involve cache aware/cache oblivious algorithms. More research is needed on virtual memory, file caching, and latency hiding. For example, can we build programming languages/ hardware with first- class support for hierarchical data structures? Will streaming models be a better way to design some HEC software? How can we make more efficient multi-threaded software/hardware?

6.4. Best Practices and Education

As our applications become more interdisciplinary, better education is becoming essential for the effective use of HEC systems. Ideally, application scientists would not need to be experts on parallel programming because a multidisciplinary team would include an expert on HEC performance programming and an expert on modern software engineering practices. However, we do not have the trained people to make this commonplace.

Computer science students need to be motivated to learn that performance is fun and application scientists need more training in software life-cycle management. In general, both educators and students need more instructional access to HEC systems. A program to increase support for student fellowships in high-end computing is essential.

7. PERFORMANCE MODELING, METRICS, AND SPECIFICATIONS

David Bailey, Chair
Lawrence Berkeley National Laboratory

Allen Snavely, Vice Chair
San Diego Supercomputer Center

The single most relevant metric of high-end system performance is *time to solution* for the specific scientific applications of interest. Reducing the time to solution will require aggressive investment in understanding all aspects of the program development and execution process (programming, job setup, batch queue, execution, I/O, system processing, and post-processing).

The current practice in system procurements is to require vendors to provide performance results on some standard industry benchmarks and several scientific applications typical of those being run at the procuring site. Constructing these application benchmarks is both cost- and labor-intensive, and responding to these solicitations is very costly for prospective vendors. Moreover, these conventional approaches to benchmarking will not be suitable for future acquisitions, where the system to be procured may be more than ten times more powerful than existing systems.

Improved community-standard benchmarks would be welcome. Both large-scale and low-level benchmarks would help to streamline and consolidate procurements. Looking to the future, performance modeling looks even more promising. Recent successes suggest that it may be possible to accurately predict the performance of a future system, much larger than systems currently in use, on a scientific application that is much larger than any currently being run. However, significant research is needed to make these methods usable by non-experts.

Another idea that has considerable merit is to exploit the highly parallel systems available at many research centers to perform simulations of current and future high-end systems. This is now possible in the wake of recent developments in parallel discrete event simulation that permit simulations of this type to be done in parallel. One particularly compelling application of such technology is simulating the operation of very large inter-processor networks for future high-end systems.

Research is also needed to bolster the capabilities to monitor and analyze the exploding volume of performance data that will be produced in future systems. On-the-fly reduction of performance trace data, as well as intelligent analysis of this data, will significantly enhance the utility and usability of these performance tools. All of this research will require significant involvement by vendors, and thus some dialogue will be needed to resolve potential intellectual property issues.

7.1. Basic Metrics

The consensus of the working group is that the single most relevant metric of high-end system performance is *time to solution* for the scientific applications of interest. Time to solution is comprised of several factors, including: 1) time devoted to programming and tuning; 2) problem set-up time and grid generation; 3) time spent in batch queues; 4) execution time; 5) I/O time; 6) time lost due to job scheduling inefficiencies, downtime, and handling system background interrupts; and 7) job post-processing, including visualization and data analysis.

Although our primary focus is on execution time, we emphasize that the other items above are significant and cannot be ignored. Indeed, all of these components of the solution time must be reduced if we are to utilize future systems effectively. For example, significantly increasing the computational power of a

system will not be very beneficial if the time required for writing the results to disk dominates the total execution time. Such considerations underscore the need for "balanced" systems, although no one "balance" formula can suffice for all applications.

Programming time is a very significant issue. For some high-end computing applications, software costs constitute as much as eighty percent of total system cost (software plus hardware plus maintenance). This is due in part to the heavy reliance on the message passing interface (MPI) programming model, which, although generally superior in delivering good execution performance, is widely regarded as rather difficult to use. Partly because of this difficulty, technical software firms have been reluctant to port their programs to highly parallel platforms; as a result, relatively few large corporations have exploited high-performance computing technology.

The first step toward improving this state of affairs is to identify and measure the key factors that make high-end programming difficult, including programming models, language, level of abstraction, and barriers to re-use. Such metrics would be useful for determining software investment strategies, as well as in gauging progress in the area of languages, compilers, and tools. Currently, there are no good metrics for assessing programming difficulty. Programming difficulty for conventional systems and languages has been studied in the software engineering community; perhaps some of these methodologies could be applied in the high-end arena.

Closely related to the question of programming difficulty is the issue of time spent tuning to achieve good performance. Currently, there are no objective measures of the effectiveness or ease of use of tuning tools, as far as we are aware. This issue is addressed in detail below.

Some work has been done in measuring system-level efficiency. There are significant differences in efficiency between some existing systems, exhibited by the fact that often only 75 percent or so of the available processor-hours in a given week's operation are spent executing user jobs, even though the batch queues have jobs ready to execute. One relevant study is the Effective System Performance (ESP) benchmark [22], which measures several system-level factors, including efficiency of the job scheduler, job launch times, effectiveness of checkpoint-restart facility (if any), and system reboot times.

We emphasize that many scientists have long complained that their jobs spend inordinate amounts of time in batch queues. This time is just as unproductive as the time spent waiting on a job because of poor execute-time performance. These considerations underscore the importance of focusing resources on those scientific projects most worthy of this valuable resource, and avoiding the temptation to over-subscribe systems. It also underscores the importance of efficient job schedulers and facilities such as system-level checkpoint-restart.

For high-end computing, the primary metric of interest is the execution time for key scientific applications. Using this metric avoids many of the common pitfalls in performance reporting, such as the use of less-than-optimal algorithms to exhibit a floating-point operation per second rate that is superficially high. When reporting computation rates, responsible scientists typically use an operation count formula based on the most efficient practical algorithms known for the target application. Thus, if a less-than-optimal algorithm is selected for some reason, the reported performance will not be distorted by this selection. This is simply a restatement of the principle that time to solution is the best figure of merit.

However, no one single figure of merit and no one definition of system balance can encapsulate the full value of a high-end system. At the very least, important decisions such as procurements should be based on several benchmarks, reflecting the breadth of the scientific applications for which the system is targeted.

7.2. Current Practice in System Procurements

Currently, high-end system procurements at federally funded research centers are handled by a process that includes numerous general hardware and software specifications, performance results on some standard community benchmarks (e.g., the Linpack or SPEC benchmarks), and timings or related performance figures on a set of scientific programs thought to be typical of the actual applications programs run at the center. (See chapter 9 for a more detailed discussion of procurement strategies and recommendations.) Composing the list of specifications and constructing the set of benchmarks is a highly labor-intensive process. Often it is necessary to make significant changes to the application benchmarks, which are obtained from the scientists and engineers using the system, to ensure standard conformance and portability.

The specifications and benchmarks are usually selected with the goal of acquiring a system that is more powerful than the systems currently in operation. Application benchmarks are validated on existing systems. In addition to being a dubious means of projecting the future full-system performance, such benchmarks typically are not very effective in disclosing system difficulties that only arise when the entire system is devoted to a single computational job.

Once proposals have been received from prospective vendors, the review committee typically evaluates them using a process that involves factors such as performance, suitability for the mission, floor space, and power requirements, as well as total cost. Indeed, the selection process is akin to the constrained optimization problems studied in the field of operations research, although no one, to our knowledge, has yet formally applied techniques of operations research to a system selection.

Prospective vendors must devote considerable resources to responding to the requests for proposals for high-end systems, and to the benchmarks in particular. Moreover, the current lack of any standard, discipline-specific benchmarks across centers means that vendors cannot amortize costs for porting and tuning codes across multiple solicitations. As a result, the prices of high-end systems quoted to federally funded research centers must be increased to recover solicitation response costs. These high costs often discourage smaller companies from attempting to compete in high-end solicitations.

In short, while prevailing procurement practices are usually effective in selecting systems that are reasonably well designed, the process is lengthy and expensive both for the government and for participating vendors. Thus, any technically valid methodologies that can standardize or streamline this process will result in greater value to the federally funded centers, and greater opportunity to focus on the real problems involved in deploying and utilizing high-end systems.

If any of the current initiatives to revitalize high-end computing are successful, then federally funded high-end computing centers will be faced with the challenge of acquiring systems that are potentially ten or more times more powerful than any system currently fielded. With such a large gap between present and future systems, the applicability of conventional benchmarking methodology will be questionable. Novel architectures, distinct in design and technology from any existing systems (e.g., those being explored in DARPA's HPCS program), will compound this challenge. In short, a significantly improved methodology for system selection will be required for future solicitations.

7.3. Performance-Based System Selection

The field would benefit from one or more community-standard benchmarks to complement the scalable Linpack benchmark that has been used for many years. The longevity of the Linpack benchmark is a tribute to its thoughtful design because it can be arbitrarily scaled in size to match the increased capabilities of modern high-end systems. On the other hand, many believe that that this benchmark emphasizes performance on regular, dense matrix operations that possess strong data locality, and ignores

the realm of irregular, sparse data access typical of the majority of modern scientific computing applications.

Thus, there is a strong need for one or more new community-standard benchmarks that characterize other important aspects of high-end scientific computing, yet share with the scalable Linpack benchmark the desirable characteristics that have made it such a success. Some discipline-specific, community-standard benchmarks would also be valuable, as they would enable vendors and computing centers to consolidate their benchmarking efforts, and would also facilitate useful interdisciplinary performance studies.

A set of standardized low-level benchmarks would also be valuable. The objective would be to define a modest-sized set of benchmarks that collectively characterize most of modern high-end computing, and possibly enable future high-end computing requirements to be defined as a combination of these benchmarks.

7.3.1. Performance Modeling

The best possibility for better understanding performance phenomena, and for assisting in intelligent system selection, may lie in performance modeling. As one example, accurate performance models have been developed for several full applications from the DOE ASCI workload [8, 11, 12], and these models are routinely used for system design, optimization, and maintenance. Moreover, a similar model has been used in the procurement of the ASCI Purple system, predicting the performance of the SAGE code on several of the systems in a recent competition [16]. Alternative modeling strategies have been used to model the NAS Parallel Benchmarks, several small PETSc applications, and the applications POP (Parallel Ocean Program), NLOM (Navy Layered Ocean Model), and Cobal60, across multiple compute platforms (IBM Power3 and Power4 systems, a Compaq Alpha server, and a Cray T3E-600) [4, 17]. These models are very accurate across a range of processors (from 2 to 128), with errors ranging from one to sixteen percent.

These results suggest that it is possible to accurately predict the performance of a future system (much larger in size and employing a distinct design from hardware currently in operation) [20], running a future scientific application (much larger in problem size than currently being run). We can even envision that a future call for proposals will specify that the vendor provide results on a set of low-level “atomic” benchmarks, generating the required input data, for performance models of key applications. Decision makers would then have not only performance data but also the capability to pursue various “what if” scenarios. Other uses include improved system configuration and system maintenance [4, 8, 9, 11, 17].

Executable analytical performance evaluation also shows promise [18]. This methodology can evaluate early-stage architecture designs over a wide operating range, and can aid in identifying advantageous architectural features before instruction set architectures are firmly established and system software (runtime systems or compilers) is available.

Performance models can be used within a user code to control the execution dynamically for best performance. Some researchers are considering using simple performance models to improve load balancing in unstructured grid applications. As another example, computational chemistry researchers are developing techniques to create highly accurate electronic structure codes based on specific characteristics of the system being used. In data-intensive applications, the allocation of data between memory and disk, or between local disk and global disk, can be decided based on system characteristics.

All of this underscores the need for a variety of performance modeling methodologies, ranging from simple, curve-fitting approaches to sophisticated tools that perform a thorough inventory of all operations performed by the target application program on a particular system. However, much work is required to further automate and reduce the complexity and cost of the modeling work. In addition, more work is

needed to define a better interface between "traditional tools" (such as profilers, timers, and hardware performance monitors) and modeling tools.

7.3.2. System Simulation

A few simulations have been employed in the research community to study particular aspects of system performance [16], and vendors often develop near-cycle-accurate simulators as part of their product development. However, computational scientists rarely use such tools to understand or predict the performance of their applications. Several challenges must be overcome for such simulations to be useful in understanding application performance. Perhaps most importantly, the simulation times required to analyze the performance of even a small loop are very large; the analysis of a full-length application code has been prohibitive. Another common weakness of these simulations is that they typically target only single-processor systems or, at best, shared-memory multiprocessor systems.

With the emergence of highly parallel computing platforms, we can consider highly detailed parallel simulations of scalable systems. In many applications, low-level, processor-memory behavior can be largely decoupled from the analysis of inter-processor network phenomena. Once the communication behavior of an application has been profiled, its inter-processor network behavior can be simulated by generating a sequence of communication operations that mimic the statistics of frequency and message length typical of the program's phases. One important factor is the recent development of parallel discrete event simulation (PDES) techniques [7, 10]. These techniques, such as "optimistic" speculation, enable simulations to be performed in parallel; otherwise it is very difficult to achieve even modest speedups due to the fundamental shortage of concurrency and the need for frequent, low-level synchronization.

Ideally, we envision an open-source architectural simulation framework and application programming interface that enables plug-and-play functionality between separately developed simulators for different architectural features (e.g., processor-in-memory, polymorphic multithreaded processor, and network), and would also enable zoom-out and zoom-in between statistically based and cycle-accurate simulation techniques. This framework will, however, require significant advances in simulation methodologies to support concurrent use of modules running at different time scales and based on different simulation techniques. Some discrete-event simulation packages are available in the research community. It is not clear whether any of these could be adapted for the requirements described here, or whether a completely new simulation package would have to be written.

7.3.3. Performance Monitoring Infrastructure

Informal approaches to parallel performance monitoring and performance data analysis may be currently acceptable; however, such approaches will be inadequate once systems are fielded with multiple levels of parallelism throughout the system's compute nodes, I/O system, network, and memory hierarchy, and once they include tens or hundreds of thousands of compute nodes. It is also unlikely that novel architectures can be effectively modeled and utilized without an advanced monitoring infrastructure.

Advanced facilities for hardware performance monitoring will be required to obtain performance data without significant perturbation. A key challenge, beyond counting events throughout the system, is in gathering and interpreting the exploding quantity of data. Today, collecting memory access pattern data, which is often crucial for understanding performance on deep-memory-hierarchy machines, implies a slowdown of three orders of magnitude [26]. However, many applications of interest run for hours or days, during which their performance behavior changes frequently. Systems with tens or hundreds of thousands of processors will greatly compound this problem of performance data analysis.

Several alternatives are being explored, ranging from clever statistical sampling schemes to on-the-fly analysis of performance data that would reduce the amount of data involved. Meaningful analysis of this data will require advanced techniques such as multivariate statistical methods [1], knowledge discovery

tools [28], time series analysis [30], and advanced visualization schemes [2] to distill important facts from these potentially massive data sets. This analysis can then be used to select the key features to apply to performance monitoring and to build predictive models of the performance of a single processor as well as the entire parallel system.

A unique opportunity exists for performance researchers to work with vendors to improve the selection of hardware performance data. Ideally, the design of performance monitoring hardware should be driven by the data input needs for application performance modeling and analysis, rather than modeling and analysis capabilities being limited by the available data. For example, one key item that current hardware monitors lack is information regarding memory addresses, such as data on gaps or patterns between successive addresses. This data would provide valuable insights into the memory behavior of a user program. We hope that vendors will consider counters useful to application developers and performance tuners as well, for example by implementing the PAPI proposed standard metrics [3].

Another area where the performance research and vendor design communities could collaborate is enhancing the performance monitoring facilities of inter-processor network hardware. Although network hardware often includes some performance monitoring facilities, the inability to associate performance data with a specific application code significantly hinders applying the data to application performance evaluation. The use of reconfigurable technology (e.g., FPGAs) might be of use to support performance monitoring applications for both hardware engineers and end-users. Determining what events are most important to monitor, designing systems to support low-overhead monitoring that generates data useful to application developers, and designing software to utilize this information are important topics of future research.

Any improvements in the capabilities of performance tools must be matched by a corresponding improvement in ease of use; otherwise they will have only limited impact in the overall goals of reducing time to solution and simplifying system acquisitions. We envision a set of standard templates for performance analysis that automatically engage typical performance analysis scenarios, using advanced tools. High-level tools could also increase the user base of performance facilities by applying techniques of automatic knowledge discovery to performance data. The application of techniques such as decision trees to performance data has been initially explored [19, 14], but significant additional research is needed in this area.

It is also important that high-end computing centers make a commitment to port their performance tools to new systems, and keep such software up to date. If users cannot be assured that these tools will continue to be supported over multiple generations of hardware, they are unlikely to make the investment of time and effort to use them.

7.3.4. Libraries, Compilers, and Self-Tuning Software

It is not sufficient to merely study the performance of large future systems; facilities for automatic and/or semi-automatic performance tuning must also be improved. One possible approach is to expand the scope of optimized scientific libraries for high-performance computing. Three canonical examples are the ScaLAPACK, PETSc, and the NWChem libraries. Some related efforts include the emergence of the Community Climate Model (CCM) in the climate-modeling community, and similar efforts to unify fusion and accelerator-modeling computations.

One of the more promising developments is the recent emergence of “self-tuning” library software. Examples include the FFTW library [11] and versions of ATLAS, ScaLAPACK, and LFC library routines [10]. In an initialization step, a program first tests different computational strategies (such as different parameters for array padding and cache blocking). The tuning program then selects the option that demonstrates the best performance for future production runs. This general approach can be extended

to almost any large-scale software library. However, devising tests, determining optimal parameters, and using the resulting parameters in the production code must be simplified if this general scheme is to be implemented widely. One possibility combines rapid, on-the-fly performance modeling with such self-adaptive, self-tuning codes to narrow the parameter space for trying different computational strategies.

Eventually, these self-tuning facilities can be incorporated directly into conventional user code. We foresee the time when self-tuning facilities will be understood well enough that they can be inserted by a preprocessor (and eventually perhaps by a compiler) into a user code at the start of the main program, or even at the subroutine level. The basic facilities have already been demonstrated in current research, including self-tuning library software, performance assertions, compiler enhancements, and semiautomatic code modifications [23].

It is instructive to recall the history of vector computing. Initially, compilers offered little or no assistance; it was necessary for programmers to explicitly vectorize loops. Then semi-automatic vectorizing compilers became available, which eventually were quite successful. The final step was runtime vectorization, with compilers generating both scalar and vector code, and then deciding at runtime if the vector code were safe or more efficient. There is a similar long-term potential for self-tuning code that exploits performance monitoring. Other ideas for compiler technology that show promise include dynamic compilation and compile-time searching for optimal run-time alternatives, including array blocking, loop fusion and fission, flexible data layout, and array padding. Since these changes in several cases go beyond the limits of what is permissible according to existing language standard definitions, this points to the need to work with language standard committees in tandem with this research.

8. APPLICATION-DRIVEN SYSTEM REQUIREMENTS

Michael Norman, Chair
University of California at San Diego

John Van Rosendale, Vice Chair
Department of Energy

During the working group's discussions, researchers presented the needs of quantum chromodynamics (QCD), fusion research, computational chemistry (including environmental and catalysis), and the biological sciences. Working group participants provided additional requirements for the following disciplines: accelerator physics, astrophysics and cosmology, aviation, atmospheric science, geophysics, materials science, and nanoscience. With this background, the working group considered the computing system needs and requirements for next-generation computational science.

8.1. Application Challenges

All of the disciplines mentioned above are large users of current high-end systems, and will continue to be in the coming decade. Computational QCD sets the scale of what may be considered large usage today. In aggregate, the U.S. QCD research community sustains 0.5-1 teraflops/second (TF) on all resources available to it. A state-of-the-art calculation consumes 0.8 petaflop-hours or $3x10^{18}$ floating point operations. Currently, we conservatively estimate that computational chemistry consumes about 50 TF on a 24/7 basis. Other large users are within an order of magnitude of this number.

Representatives of multiple disciplines at the workshop made the quantitative case for speedups in sustained performance of 50 to 100 over current levels to reach new, important scientific thresholds. In QCD, architectures with a sustained performance of 20 to 100 TF would enable calculations of sufficient precision to serve as predictions for ongoing and planned experiments. In magnetic fusion research, sustained execution of 20 TF would allow full-scale tokamak simulations that resolve the natural length scales of the microturbulence responsible for transport, as well as enable self-consistent, gyrokinetic modeling of the critical plasma edge region. Although the needs of *ab initio* quantum chemistry simulation for industrial and environmental applications are almost limitless, 50 TF was identified as an important threshold for developing realistic models of lanthanides and actinides on complex mineral surfaces for environmental remediation, and for developing new catalysts that are more energy efficient and generate less pollution.

To demonstrate the range of application needs, we consider two examples: lattice QCD and computational biosciences.

8.1.1. Lattice QCD

Recent advances in algorithms, particularly new formulations of QCD on the lattice, now enable calculations of unprecedented accuracy, provided the required computational resources are available. Moreover, lattice gauge theory was invented in the United States, and U.S. physicists have traditionally been intellectual leaders in the field. However, for the past five years, greater investments in computational resources have been made in Europe and Japan.

Within the next year, European lattice theorists will have dedicated, customized computers that sustain well over 15 TF. If U.S. physicists are to regain leadership of the field and be able to attract outstanding young scientists, comparable resources are needed. Within the next five years, the U.S. lattice community needs to sustain hundreds of teraflops/second and, by the end of the decade, multiple petaflops/second. Simplifying features of QCD simulations provide a pathway for doing so through the construction of special purpose hardware.

8.1.2. Computational Biosciences

High throughput technologies are revolutionizing the way biologists are gaining an understanding of how cells and organisms behave. These new technologies, which generate massive amounts of data, are enabling the study of the complexity of living systems, and are leading to a paradigm change in the way biological research is conducted—a transition from the reductionist molecular level to systems biology. Computational sciences and large-scale computers will play a key role in this transition. If the biology data tsunami is not addressed, biologists will not be able to extract the full range of possible insights from the data.

Examples of high throughput biological sources include genomic data for multiple organisms and for individuals, protenomic data from mass spectrometry and arrays (~100 GB/day/mass spectrometer and for 50 mass spectrometers, 5 TB/day) and cell imaging (a FRET analysis of a cell generate a megapixel/millisecond). (There are many such spectrometers, leading to tens of TB/day.) Likewise, cryo-electron microscopy for complex cellular and molecular structures (50 to 100 GB/day/spectrometer), x-ray imaging at synchrotrons (TB/day), and molecular dynamics simulations (2.5 TB/day on a 10 TF computer) are generating large data sets. These data sets not only need to be stored, but must be made readily available to a broad community that places real demands on networks.

As experimental systems in biology become more expensive to purchase and maintain, they must be localized, which requires real-time, remote control of experiments. In turn, this requires significant networking resources for user access. In this distributed resource model, each experiment transmits data back to the user, which enables the user to make decisions about experiment control. The amount of data transmitted *by* the user is usually small, but a high-integrity network is needed with low latency. The amount of data transmitted *to* the user may be much larger, making bandwidth and latency management important for real-time control.

8.2. System Challenges

These two examples are illustrative—members of every discipline present at the workshop cited the difficulties in achieving high, sustained performance (relative to peak) on complex applications as the key hardware challenge. This reflects the imbalance between processor speed and memory latency and bandwidth. A more serious imbalance exists between interprocessor latency and bandwidth.

Application scientists have invested considerable effort in optimizing parallel applications for current architectures. Many researchers have reported application scaling to several thousand processors for fixed work per processor experiments (weak scaling). Fixed-size problems, such as occur in molecular dynamics and climate modeling, encounter scaling barriers at much lower processor counts, depending on the size of the problem (strong scaling.) Applications with a single, well-optimized kernel, such as lattice QCD, do well on current architectures, although even here a performance sacrifice of a factor of 3 to 4 exists due to inadequate memory bandwidth, except on computers specially customized for this problem.

The key challenge is the difficulty building and maintaining complex application software. The single programmer model is not sustainable, and a multidisciplinary team approach is not only desirable but also essential, given the level of complexity present in modern high-end applications. Although new and efficient algorithms are also needed, the key algorithms challenge is not about algorithms *per se*, but finding ways to integrate models at different length and time scales into holistic, multi-scale simulations. (See section 6.1 for a software perspective on this problem.)

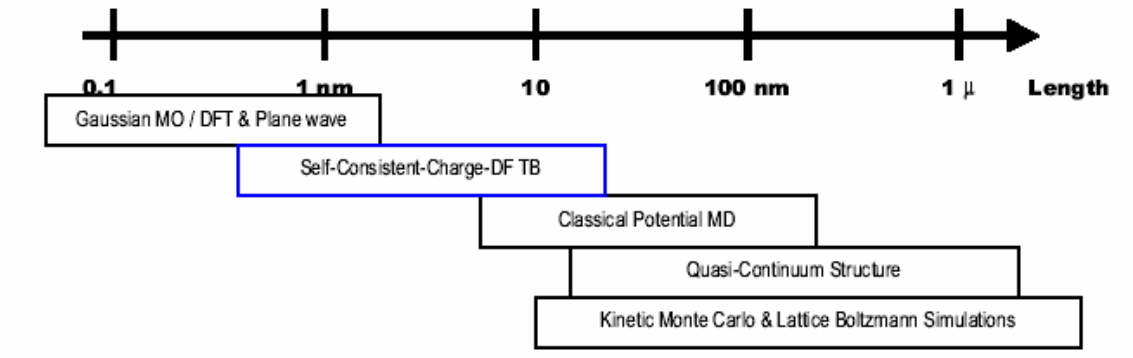

Figure 8.1 Multi-Scale Computational Nanoscience Simulations (Source: M. Gutowski)

Figure 8.1 shows an example of the importance of multi-scale simulations from nanoscience. Each box represents an algorithm or application developed to model phenomena on the length scale given by the axis. To fully understand systems of interest, we must integrate models across the entire range of length scales.

8.3. Current System Limitations

Representatives of every discipline who attended the workshop cited inadequate memory bandwidth as a critical limitation. Most disciplines also cited inadequate inter-processor communication latency and bandwidth as critical system parameters, although there was considerable dispersion in how much improvement is needed.

The lattice QCD community has carefully studied the relation of communication capabilities to application performance. QCD applications employ a simple 4-D block decomposition of the simulated 4-D space-time volume. To fully overlap communication with computation, an inter-processor communication speed of 0.364 MF/L MB/s per processor is needed, where MF is the sustained execution speed in megaflops/second of the QCD kernel per processor, and L^4 is the number of cells in one block. Given that most of the communication is nearest neighbor in this application, this number serves as a lower limit for more communication-intensive applications, such as those with global elliptic solvers like cosmology and material science.

Comparing application performance on a single processor versus on many processors provides a rough measure of the imbalance between processor and communication speeds on current architectures. The disciplines surveyed report a degradation of performance in the range of 2 to 10. Multiplying this by a typical factor of three required to achieve 50 percent of peak on a single processor shows that the loss of productivity is somewhere in the range 6 to 30, depending on application.

8.4. Support Environment Requirements

Applications have become so complex that multidisciplinary teams of application and computer scientists are needed to build and maintain them. The traditional software model of a single programmer developing a monolithic code is no longer relevant at the high end and cutting edge. In particular, large teams with diverse domain expertise are needed to integrate multi-scale simulation models. No single person or small group has the requisite expertise. This need is particularly acute in the magnetic fusion simulation community, where dozens of component models have been developed for different pieces of the problem. New team structures and new mechanisms to support distributed collaboration are needed, since the intellectual effort is generally distributed, not centralized.

The data tsunami (i.e., the flood of both input and output data for HEC systems) is a second challenge. Currently, terabytes per day of experimental data are being collected and archived at HEC centers for biological/biomedical research, weather prediction, high energy physics, earth and space science, and other uses. This number is limited mainly by detector technology and network communication bandwidths. Flagship projects such as NASA's Earth Observing System, DOE's Genomes to Life, and NSF's National Virtual Observatory anticipate data collection rates growing by at least two orders of magnitude by decade's end. Many of today's high-end numerical simulations also produce terabytes per run spread over many days. With the anticipated increases in computer capability discussed here, individual simulations will produce terabytes per day within five years.

Massive, shared-memory architectures with greater I/O capabilities are needed for data assimilation, analysis, and mining. The motivation for shared-memory systems is that many data ingest or analysis codes are sequential, legacy codes. Although these codes could be parallelized, developing and validating new implementations is often perceived as a poor use of human resources. The capabilities of such shared-memory systems must match the capabilities of the HEC systems they serve. A poll of working group participants suggests that a data analysis server should have no less than one-quarter of the memory and compute capacity as the full HEC system, and roughly the same I/O bandwidth to mass store.

Establishing computational "end stations" is one integrated solution to the needs of developers and users. This is an analogy to the organization of high-end experimental facilities (see Figure 8.2 for an

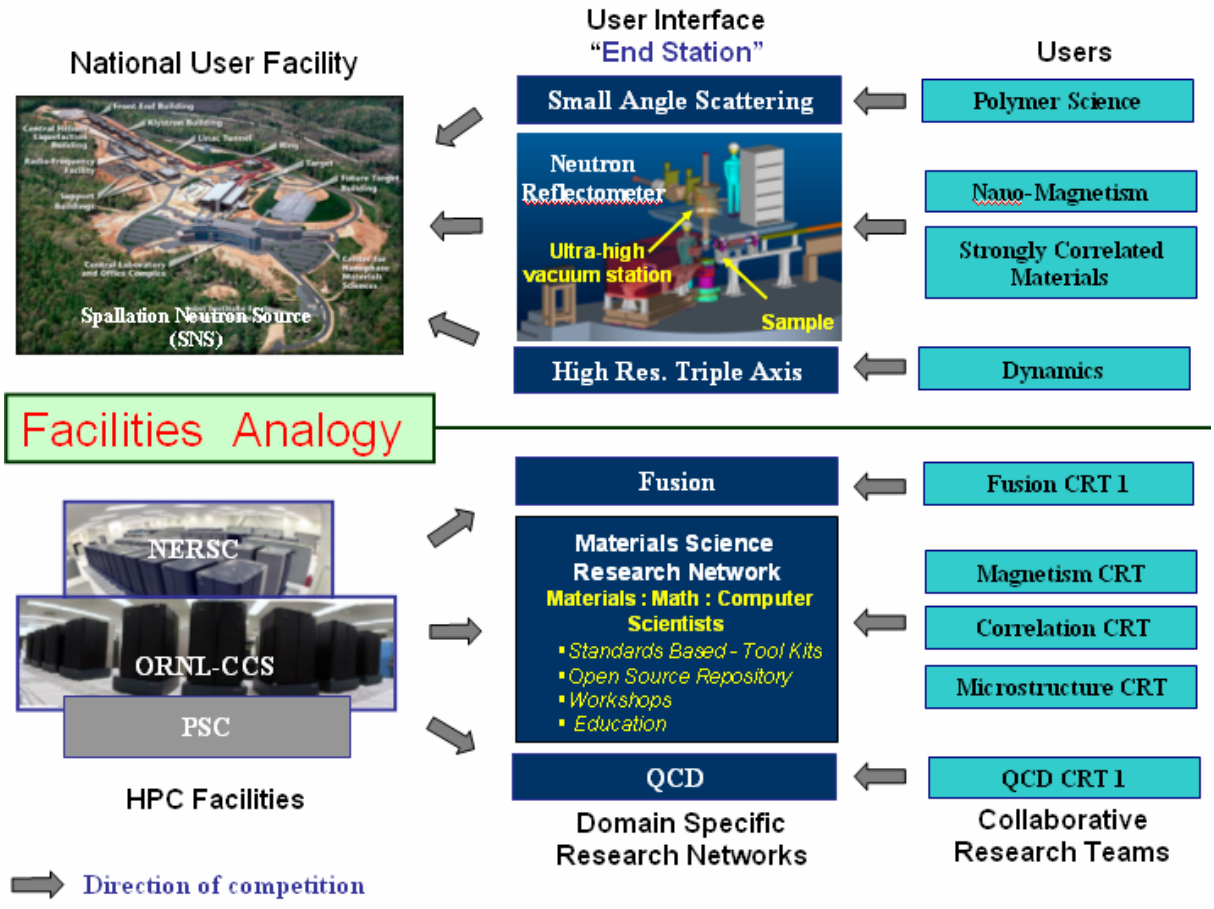

Figure 8.2 Facilities Support for Computational Science

illustration). In experimental science, users interface with a national user facility via end stations that house instruments and their associated instrument specialists.

Application codes and their associated analysis tools are the instruments of computational science. The developers of these applications can be likened to instrument specialists, in that they possess the most detailed knowledge of the applications' capabilities and usage. Unlike experimental science, however, the computational end stations need not be located at the HEC facilities. In fact, they need not be centralized at all, but may be distributed research networks, as are emerging in many fields. Within this facilities analogy, HEC users in the form of collaborative research teams would interface primarily with the application specialists within domain-specific research networks that develop, optimize, and maintain the relevant applications.

9. PROCUREMENT, ACCESSIBILITY, AND COST OF OWNERSHIP

Frank Thames, Chair
NASA Langley Research Center

James Kasdorf, Vice Chair
Pittsburgh Supercomputing Center

The requirements specifications, along with establishment and application of sound and precise evaluation criteria and the selection of appropriate contract vehicles, are the most important elements of system procurements. Each is discussed in some detail below, along with some comments on improving the overall acquisition process and some miscellaneous items.

9.1. Procurement

9.1.1. Requirements Specification

The most important specification is to elucidate precisely the fundamental science requirement(s) the proposed system must satisfy. This is key to a vendor's understanding of the computer system requirements and details of the desired application environment. Prospective vendors should be encouraged to hold discussions with the intended user community to gain a full understanding of the desired environment.

The specification of leading-edge, high-end computing systems is not an exact science, and it should not be treated as such. The specifications should strongly emphasize functional requirements rather than detailed technical implementations. Similarly, one should avoid over-specifying the requirements for advanced systems. Because the desired system may not exist at the time the acquisition process is initiated, mandatory requirements should be chosen with great care or, preferably, replaced with requirements weighted to convey their relative importance. Similarly, procurements should consider permitting reasonable flexibility in specifying delivery dates.

Because vendors often reply to multiple solicitations simultaneously, overly aggressive delivery schedules may force some vendors to no-bid a solicitation. Federal solicitations require competition. Thus, it is unwise to cite requirements that may limit, or even eliminate, competition. The best procurements are those that draw the maximum participation from the HEC vendor community. Because these procurements deal with very advanced systems, it is wise to employ flexible, and even novel, acquisition approaches. For example, the use of vendor/customer technical partnerships is particularly appropriate, as is the use of contract options to permit flexible deliveries. Lastly, one should consider carefully the fundamental differences in the specifications for systems with diverse uses (e.g., capability or capacity uses).

9.1.2. Evaluation Criteria

Evaluation criteria dominate acquisitions because they define customer priorities; one of these critical criteria is cost. The appropriate cost metric might be that of *Total Cost of Ownership*—that is, the total life-cycle cost of acquiring, installing, operating, and disposing of large HEC systems. Suggested elements of the total ownership cost are covered in detail below.

From a technical standpoint, it is paramount that "real" benchmarks be used to categorize system performance. This is not the simplest of tasks even for short-term contracts. For longer-term procurements where the "real" benchmarks may not be known, the customer should specify required application speedup. This provides flexibility, as customers often can project the general level of increase in application complexity.

As discussed in chapter 7, benchmarking can be expensive for vendors, so it is best not to push very hard on difficult benchmarks. Should the expense become too great, some vendors may choose not to participate in a given acquisition opportunity. As HEC systems become larger and even more complex, a composite of application benchmarks may be inadequate to accurately measure system performance. This implies that benchmarks must be rethought to ensure that the desired system is acquired. Because high-end systems are often difficult to specify, and they occupy a small market segment, customers should use a best-value approach in their evaluations (i.e., a balance between cost of ownership and technical performance).

Finally, as in all acquisitions, risk should be evaluated. Risks take multiple forms, and the ones cogent to the particular acquisition should be considered. Risk types include schedule, technical, and cost. Evaluating risk is not easy and is often somewhat subjective. Nevertheless, it is very important, particularly for HEC systems.

9.1.3. Contract Type

Most government and private entities use multiple contract types that depend on many factors. However, they are normally tailored to both requirements and market diversity. For long-term, leading-edge HEC procurements, a cost-plus contract format is preferred. This is particularly true for contracts that are developmental (i.e., contracts where it is difficult to document and categorize all risks). The General Services Administration provides a variety of contracts that any federal agency can use. The use of existing vehicles can lead to substantially lower life-cycle costs; full-blown, competitive procurements are time-consuming and, hence, quite costly.

9.1.4. Process Improvement

The working group identified four cogent elements in process improvement. First, HEC procurers should consider employing the "DARPA Process" for HEC acquisitions. This process has two principal features:

- It has an "R&D" flavor; that is, it is low on detailed specifics and long on desired outcomes.
- It features multiple awards and down selects to move from development to prototype and/or production systems.

The second process improvement concerns acquisition schedule. Because both customers and vendors are likely to be engaged in multiple acquisitions, all parties must attempt to adhere to a rigid schedule. This not only lowers cost, but also promotes maximum vendor participation. The third suggestion was that the customers engage users in the acquisition process from the very earliest stages. This reduces vendor risk and provides users with "decision proximity." Finally, it is strongly suggested that open communications between customers and vendors be maintained for as long as legally practical and openly fair.

9.1.5. Other Considerations

There was unanimous agreement among the working group that the federal government should not employ a single acquisition for all federal HEC systems. This would lead to a user disconnect and lead to the inevitable "Ivory Tower" syndrome that is counterproductive in a development-oriented environment like HEC. The lesson is not to "over-centralize." Yet another concern was inconsistency in the manner that government procurement regulations are interpreted and implemented, even within the same agency. Such inconsistencies lead to general confusion, longer schedules, and increased cost.

The working group was asked to comment on any factors they felt were important in the revitalization of the HEC industry. The group provided two summary recommendations:

- The market should be composed of multiple vendors (greater than three), each with an annual business base in excess of several hundred million dollars. This business base is necessary to maintain

the high level of technical investments needed to remain competitive in the technically sensitive HEC market.The HEC market is not large (~$1B/year); however, to remain commercially viable, these companies must have adequate returns.

9.2. Accessibility

The working group generally felt that accessibility was not a compelling issue. In summary, the findings were that there are many standard vehicles to use to implement interagency agreements to provide HEC accessibility

- *Suggested process:* Current large suppliers (e.g., NSF, DoD HPC Modernization) would add additional capacity to existing sites to service the needs of agencies with smaller requirements that are insufficient to justify large expenditures for a sustaining HEC infrastructure.
- *Implementation suggestion*: Consider a single point of contact for federal agencies to go to for accessibility suggestions (not implementation). The NCO would be a logical choice as they are cognizant of federal IT R&D.
- *Critical issue:* The requiring agency must have *designated, multiple-year funds* to purchase HEC computational capabilities from the large suppliers. HEC capability cannot be purchased "by the yard." There must be sustaining commitments.

9.3. Cost of Ownership

The working group identified several factors related to total cost of ownership. For easy reference, these factors are consolidated in Table 9.1.

One item in Table 9.1 related to lost opportunity costs requires some elaboration. These costs are difficult to quantify and anticipate; however, because they can be high, they should be given careful thought. Although lost opportunity costs vary significantly, here are some suggestions for the costs being considered:

- Lost research opportunities due to excessive downtime or inappropriate system architecture.
- Low user productivity due to lack of application programming tools.
- Legacy codes not optimized for "new" system architecture.

Overall, one can attribute many of these costs to the substantial difficulty in evaluating system and application support tools, which affects productivity for application code development as well as production computing.

Factor	Type	Comments
Procurement of Capital Asset	One-time	
- Hardware		
- Software licenses		Implementation of procurement
- Workforce		
- Cost of money		For LTOPs
Maintenance of Capital Assets	Recurring	
- Hardware		
- Software licenses		
Services	Recurring	Workforce dominated; will inflate yearly (~4%)
- System administration		
- Application support/porting		
- Operations		
- Security		
Facility	Recurring	
- HVAC		
- Electrical power		
- Maintenance		
- Initial construction		If required
- Floor space		
Networks	Recurring	
- Local Area Network		
- Wide Area Network		
Training	Recurring	
Miscellaneous		
- Insurance	Recurring	
- Disposal of capital assets	One-time	
- Lost opportunity	Both	See text

Table 9.1 Total Cost of Ownership Factors

10. REFERENCES

[1] D. H. Ahn, J. S. Vetter, "Scalable Analysis Techniques for Microprocessor Performance Counter Metrics," *Proceedings of SC 2002*, IEEE, Nov. 2002.

[2] R. P. Bosch Jr., "Using Visualization to Understand the Behavior of Computer Systems," Stanford University Ph.D. dissertation, August 2001.

[3] S. Browne, J. Dongarra, G. Ho, N. Garner, P. Mucci, "A Portable Programming Interface for Performance Evaluation on Modern Processors," *International Journal of High Performance Computing Applications*, vol. (2000), pp. 189-204.

[4] L. Carrington, A. Snavely, N. Wolter, X. Gao, "A Performance Prediction Framework for Scientific Applications," Workshop on Performance Modeling and Analysis, ICCS, Melbourne, June 2003.

[5] B. L. Chamberlain, E. C. Lewis, and L. Snyder, "Array Language Support for Wavefront and Pipelined Computations," in *Workshop on Languages and Compilers for Parallel Computing*, August 1999.

[6] C. Chang, K. Kuusilinna, B. Richards, and R.W. Brodersen, "Implementation of BEE: A Real-time Large-Scale Hardware Emulation Engine," *Proceedings of FPGA 2003*, pp. 91-99, February 2003.

[7] A. Chauhan, C. McCosh, K. Kennedy, and R. Hanson, *"Automatic Type-Driven Library Generation for Telescoping Languages," SC03, to appear*.

[8] D. Chen, N. H. Christ, C. Cristian, Z. Dong, A. Gara, K. Garg, B. Joo, C. Kim, L. Levkova, X. Liao, R. D. Mawhinney, S. Ohta and T. Wettig, "QCDOC: A 10-Teraflops Scale Computer for Lattice QCD," *Nucl.Phys.Proc.Suppl*. 94 (2001) pp. 825-832.

[9] W. Chen, D. Bonachea, J. Duell, P. Husbands, C. Iancu, K. Yelick, "A Performance Analysis of the Berkeley UPC Compiler," *17th Annual International Conference on Supercomputing* (ICS), 2003.

[10] J. Dongarra and V. Eijkhout, "Self Adapting Numerical Algorithm for Next Generation Applications," *International Journal of High Performance Computing Applications*, vol. 17, no. 2, pp. 125-132.

[11] M. Frigo and S. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Seattle, WA, May 1998.

[12] R. Fujimoto, *Parallel and Distributed Simulation Systems*, Wiley Interscience, January 2000.

[13] J. Makino, T. Fukushige and M. Koga, "A 1.349 Tflops Simulation of Black Holes in a Galactic Center on GRAPE-6," SC2000.

[14] A. Hoisie, O. Lubeck, H. Wasserman, "Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications," *The International Journal of High Performance Computing Applications*, vol. 14, no. 4 (Winter 2000).

[15] ITRS (International Technology Roadmap for Semiconductors), http://public.itrs.net/, 2002.

[16] A. Jacquet, V. Janot, R. Govindarajan, C. Leung, G. Gao, and T. Sterling, "An Executable Analytical Performance Evaluation Approach for Early Performance Prediction," *Proceedings of IPDPS'03*, 2003.

[17] D. Jefferson, B. Beckman, F. Wieland, L. Blume, M. DiLoreto, P. Hontalas, P. Laroche, K. Sturdevant, J. Tupman, V. Warren, J. Wedel, H. Younger, S. Bellenot, "Distributed Simulation and the Time Warp Operating System," *11th Symposium on Operating Systems Principles*, Austin, TX, Nov., 1987.

[18] D. J. Kerbyson, H. Alme, A. Hoisie, F. Petrini, H. Wasserman, M. Gittings, "Predictive Performance and Scalability Modeling of a Large-Scale Application," *Proceedings of SC2001*, IEEE, November 2001.

[19] M. Mathis, D. Kerbyson, A. Hoisie, "A Performance Model of Non-Deterministic Particle Transport on Large-scale Systems," *Workshop on Performance Modeling and Analysis, ICCS*, Melbourne, June 2003.

[20] D. J. Kerbyson, H. J. Wasserman, A. Hoisie, "Exploring Advanced Architectures Using Performance Prediction," in *Innovative Architecture for Future Generation High-Performance Processors and Systems*, IEEE Computer Society Press, 2002, pp. 27-37.

[21] B. P. Miller, M. D. Callaghan, J. Cargille, J. K. Hollingsworth, R. B. Irbin, K. Karavanic, K. Kunchithapadam, T. Newhall, "The Paradyn Parallel Performance Measurement Tools," *IEEE Computer*, vol. 28 (1995), no. 11, pp. 37-46.

[22] R. Numrich and J. Reid, "Co-Array Fortran for Parallel Programming," ACM Fortran Forum, vol. 17, No. 2, pp. 1-31, 1998.

[23] D. Quinlan, M. Schordan, B. Philip and Kowarschik, M., "Parallel Object-Oriented Framework Optimization," to appear in *Special Issue of Concurrency: Practice and Experience*, 2003.

[24] M. Rosenblum, "SimOS," available at http://simos.stanford.edu.

[25] J. S. Shapiro, N. Hardy, "EROS: A Principle-Driven Operating System from the Ground Up," IEEE Software, January/February 2002.

[26] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia A. Purkayastha, "A Framework for Performance Modeling and Prediction," *Proceedings of SC2002*, IEEE, Nov. 2002.

[27] Tapus, C., I.-H. Chung, and J.K. Hollingsworth, "Active Harmony: Towards Automated Performance Tuning," in *Proceedings of SC2002*, IEEE, Nov. 2002.

[28] J. S. Vetter, "Performance Analysis of Distributed Applications Using Automatic Classification of Communication Inefficiencies," Proceedings *of the ACM International Conference on Supercomputing (ICS)*, ACM Press, 2000.

[29] J. S. Vetter, P. Worley, "Asserting Performance Expectations," *Proceedings of SC2002*, IEEE, Nov. 2002.

[30] M. Wang, T. Madhyastha, N. H. Chan, S. Papadimitriou, C. Faloutsos, "Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic, *International Conference on Data Engineering*, 2001.

[31] A. T. Wong, L. Oliker, W. T. C. Kramer, T. L. Kaltz and D. H. Bailey, "ESP: A System Utilization Benchmark," *Proceedings of SC2000*, Nov. 2000.

[32] L. Zhang, Z. Fang, M. Parker, B.K. Mathew, L. Schaelicke, J.B. Carter, W.C. Hsieh, and S.A. McKee, "The Impulse Memory Controller," *IEEE Transactions on Computers*, pp. 1117-1132, November 2001.

APPENDIX A. ORGANIZING COMMITTEE MEMBERS

Kamal Abdali, National Science Foundation
Greg Astfalk, Hewlett-Packard
Andy Bernat, Computing Research Association
Walt Brooks, NASA Ames Research Center
George Cotter, National Security Agency
Jack Dongarra, University of Tennessee
Julio Facelli, University of Utah
Stuart Feldman, IBM
John Gustafson, SUN Microsystems
John Grosh, Department of Defense
Cray Henry, Department of Defense
Rich Hirsh, National Science Foundation
Fred Johnson, Department of Energy
Alan Laub, Department of Energy
Paul Messina, Argonne National Laboratory
Jose Munoz, Department of Energy
David Nelson, National Coordination Office
Michael Norman, University of California at San Diego
Robert Peterkin, Department of Defense
Dan Reed, National Center for Supercomputing Applications (*Workshop Chair*)
Steve Reinhardt, SGI
John Van Rosendale, Department of Energy
Horst Simon, Lawrence Berkeley National Laboratory
Thomas Sterling (Caltech/JPL)
Robert Sugar, University of California at Santa Barbara
Frank Thames, NASA Langley Research Center
Gary Wohl, National Weather Service

APPENDIX B. WORKING GROUP CHARTERS AND PARTICIPANTS

The workshop was organized as a set of eight working groups, each with a specific technical charge. To balance participation across the working groups, the organizers initially assigned members to each group, based on their expression of interests. At the workshop, we allowed participants to shift groups, based on discussions at the meeting.

B.1. Enabling Technologies

B.1.1. Charter

Establish the basic technologies that may provide the foundation for important advances in HEC capability and determine the critical tasks required before the end of this decade to realize their potential. Such technologies include hardware devices or components and the basic software approaches and components needed to realize advanced HEC capabilities.

Questions:

- Provide information about key technologies that must be advanced to strengthen the foundation for developing new generations of HEC systems. Include discussion of promising novel hardware and software technologies with potential pay-off for HEC.
- Provide a brief technology maturity roadmap and investment strategy with discussion of costs to develop these technologies.

B.1.2. Participants

Sheila Vaidya, Lawrence Livermore National Laboratory *(chair)*
Stuart Feldman, IBM *(vice chair)*

Kamal Abdali, NSF
Fernand Bedard, NSA
Herbert Bennett, NIST
Ivo Bolsens, XILINX
Jon Boyens, Department of Commerce
Bob Brodersen, University of California at Berkeley
Yolanda Comedy, IBM
Loring Craymer, Jet Propulsion Laboratory (JPL)
Bronis de Supinski, Lawrence Livermore Nation Laboratory
Martin Deneroff, SGI
Sue Fratkin, CASC
David Fuller, JNIC/Raytheon
Gary Hughes, NSA
Tyce McLarty, Lawrence Livermore National Laboratory
Kevin Martin, Georgia Institute of Technology
Virginia Moore, National Coordination Office
Ahmed Sameh, Purdue University
John Spargo, Northrop-Grumman
William Thigpen, NASA
Uzi Vishkin, University of Maryland
Steven Wallach, Chiaro

B.2. COTS-Based Architectures

B.2.1. Charter

Determine the capability roadmap of anticipated COTS-based HEC system architectures through the end of the decade. Identify those critical hardware and software technology and architecture developments, required to both sustain continued growth and enhance user support

Questions

- Identify opportunities and challenges for anticipated COTS-based HEC systems architectures through the decade and determine its capability roadmap.
- Include alternative execution models, support mechanisms, local element and system structures, and system engineering factors to accelerate rate of sustained performance gain (time to solution), performance to cost, programmability, and robustness.
- Identify those critical hardware and software technology and architecture developments, required to both sustain continued growth and enhance user support.

B.2.2. Participants

Walt Brooks (*chair*), NASA Ames Research Center
Steve Reinhardt (*vice chair*), SGI

Erik DeBenedictis, Sandia National Laboratories
Yuefan Deng, SUNY at Stony Brook
Don Dossa, Lawrence Livermore National Laboratory
Guang Gao, University of Delaware
Steven Gottlieb, Indiana University
Richard Hilderbrandt, National Science Foundation
Curt Janssen, Sandia National Laboratories
Bill Kramer, NERSC
Charles Lefurgy, IBM
Greg Lindahl, Key Research
Tom McWilliams, Key Research
Rob Schreiber, Hewlett-Packard
Burton Smith, Cray
Stephen Wheat, Intel
John Ziebarth, Los Alamos National Laboratory

B.3. Custom Architectures

B.3.1. Charter

Identify opportunities and challenges for innovative HEC system architectures, including alternative execution models, support mechanisms, local element and system structures, and system engineering factors to accelerate rate of sustained performance gain (time to solution), performance to cost, programmability, and robustness.

Establish a roadmap of advanced-concept alternative architectures likely to deliver dramatic improvements to user applications through the end of the decade. Specify those critical developments achievable through custom design necessary to realize their potential.

Questions:

- Present driver requirements and opportunities for innovative architectures demanding custom design.
- Identify key research opportunities in advanced concepts for HEC architecture.
- Determine research and development challenges to promising HEC architecture strategies.
- Project brief roadmap of potential developments and impact through the end of the decade.
- Specify impact and requirements of future architectures on system software and programming environments.

B.3.2. Participants

Peter Kogge, University of Notre Dame (*chair*)
Thomas Sterling, California Institute of Technology and NASA JPL (*vice chair*)

Duncan Buell, University of South Carolina
George Cotter, National Security Agency
William Dally, Stanford University
James Davenport, Brookhaven National Laboratory
Jack Dennis, Massachusetts Institute of Technology
Mootaz Elnozahy, IBM
Bill Feiereisen, Los Alamos National Laboratory
David Fuller, JNIC
Michael Henesey, SRC Computers
David Kahaner, ATIP
Norm Kreisman, DOE
Grant Miller, National Coordination Office
Jose Munoz, DOE NNSA
Steve Scott, Cray
Vason Srini, University of California, Berkeley
Gus Uht, University of Rhode Island
Keith Underwood, Sandia National Laboratories
John Wawrzynek, University of California, Berkeley

B.4. Runtime and Operating Systems

B.4.1. Charter

Establish baseline capabilities required in the operating systems for projected HEC systems scaled to the end of this decade and determine the critical advances that must be undertaken to meet these goals. Examine the potential, expanded role of low-level runtime system components in support of alternative system architectures.

Questions:

- Establish principal functional requirements of operating systems for HEC systems of the end of the decade
- Identify current limitations of OS software and determine initiatives required to address them
- Discuss role of open source software for HEC community needs and issues associated with development/maintenance/use of open source
- Examine future role of runtime system software in the management/use of HEC systems containing from thousands to millions of nodes

B.4.2. Participants

Rick Stevens (*chair*), Argonne National Laboratory
Ron Brightwell (*vice chair*), Sandia National Laboratories

Robert Ballance, University of New Mexico
Jeff Brown, Los Alamos National Laboratory
Deborah Crawford, NSF
Wes Felter, IBM
Gary Grider, Los Alamos National Laboratory
Leslie Hart, NOAA
Thuc Hoang, Department of Energy
Barney Maccabe, University of New Mexico
Ron Minnich, Los Alamos National Laboratory
D.K. Panda, Ohio State University
Keshav Pingali, Cornell University
Neil Pundit, Sandia National Laboratories
Dan Reed, NCSA, University of Illinois
Asaph Zemach, Cray

B.5. Programming Environments and Tools

B.5.1. Charter

Address programming environments for both existing legacy codes and alternative programming models to maintain continuity of current practices, while also enabling advances in software development, debugging, performance tuning, maintenance, interoperability and robustness. Establish key strategies and initiatives required to improve time to solution and ensure the viability and sustainability of applying HEC systems by the end of the decade.

Questions:

- Assume two possible paths to future programming environments: (a) incremental evolution of existing programming languages and tools consistent with portability of legacy codes and (b) innovative programming models that dramatically advance user productivity and system efficiency/performance
- Specify requirements of programming environments and programmer training consistent with incremental evolution, including legacy applications
- Identify required attributes and opportunities of innovative programming methodologies for future HEC systems
- Determine key initiatives to improve productivity and reduce time-to-solution along both paths to future programming environments

B.5.2. Participants

Dennis Gannon (*chair*), Indiana University
Richard S. Hirsh (*vice chair*), National Science Foundation

Rob Armstrong, Sandia National Laboratories
David Bader, University of New Mexico
David Bernholdt, Oak Ridge National Laboratory
David Callahan, Cray
William Carlson, IDA Center for Computing Sciences
Siddhartha Chatterjee, IBM
Thomas Cormen, Dartmouth College

Tamara Dahlgren, Lawrence Livermore National Laboratory
Wael Elwasif, Oak Ridge National Laboratory
Thomas Epperly, Lawrence Livermore National Laboratory
Howard Gordon, Center for Computing Sciences
John Hall, Los Alamos National Laboratory
Daryl Hess, NSF
Laxmikant Kale, University of Illinois
Charles Koelbel, Rice University
Gary Kumfert, Lawrence Livermore National Laboratory
John Levesque, Cray
Lois Curfman McInnes, Argonne National Laboratory
Jarek Nieplocha, Pacific Northwest National Laboratory
Matthew Rosing, PeakFive
Surresh Shukla, Boeing
Anthony Skjellum, MPI-Software Technology
Lawrence Snyder, University of Washington
Alexander Veidenbaum, University of California, Irvine
Xiaodong Zhang, National Science Foundation
Hans Zima, NASA Jet Propulsion Laboratory

B.6. Performance Modeling, Metrics, and Specifications

B.6.1. Charter

Establish objectives of future performance metrics and measurement techniques to characterize system value and productivity to users and institutions. Identify strategies for evaluation including benchmarking of existing and proposed systems in support of user applications. Determine parameters for specification of system attributes and properties

Questions:

- As input to HECRTF charge, provide information about the types of system design specifications needed to effectively meet various application domain requirements.
- Examine current state and value of performance modeling, metrics for HEC and recommend key extensions
- Analyze performance-based procurement specifications for HEC that lead to appropriately balanced systems.
- Recommend initiatives needed to overcome current limitations in this area.

B.6.2. Participants

David H. Bailey (*chair*), Lawrence Berkeley National Laboratory
Allen Snavley (*vice chair*), San Diego Supercomputer Center

Steven Ashby, Lawrence Livermore National Laboratory
Maurice Blackmon, UCAR
Patrick Bohrer, IBM
Kirk Cameron, University of South Carolina
Carleton DeTar, University of Utah
Jack Dongarra, University of Tennessee
Douglas Dwoyer, NASA Langley Research Center
Wesley Felter, IBM

Peter Freeman, National Science Foundation
Ahmed Gheith, IBM
Brent Gorda, Lawrence Berkeley National Laboratory
Guy Hammer, DOD-MDA
Jeremy Kepner, MIT
David Koester, MITRE
Sally McKee, Cornell University
David Nelson, National Coordination Office
Jeffrey Nichols, Oak Ridge National Laboratory
Keith Shields, Cray
Jeffrey Vetter, Lawrence Livermore National Laboratory
Theresa Windus, Pacific Northwest National Laboratory
Patrick Worley, Oak Ridge National Laboratory

B.7. Application-Driven System Requirements

B.7.1. Charter

Identify major classes of applications likely to dominate HEC system usage by the end of the decade. Determine machine properties (floating point performance, memory, interconnect performance, I/O capability and mass storage capacity) needed to enable major progress in each of the classes of applications. Discuss the impact of system architecture on applications. Determine the software tools needed to enable application development and support for execution. Consider the user support attributes including ease of use required to enable effective use of HEC systems.

Questions:

- Identify major classes of applications likely to dominate use of HEC systems in the coming decade, and determine the scale of resources needed to make important progress. For each class indicate the major hardware, software and algorithmic challenges.
- Determine the range of critical systems parameters needed to make major progress on the applications that have been identified. Indicate the extent to which system architecture affects productivity for these applications.
- Identify key user environment requirements, including code development and performance analysis tools, staff support, mass storage facilities, and networks.

B.7.2 Participants

Michael Norman (*chair*), University of California at San Diego
John Van Rosendale (*vice chair*), Department of Energy

Don Batchelor, Oak Ridge National Laboratory
Bert de Jong, Pacific Northwest National Laboratory
David Dixon, Pacific Northwest National Laboratory
Howard (Flash) Gordon, National Security Agency
Maciej Gutowski, Pacific Northwest National Laboratory
Theresa Head-Gordon, Lawrence Berkeley National Laboratory
Steve Jardin, Princeton University
Peter Lyster, National Institutes of Health
Mike Merrill, National Security Agency
T.P. Straatsma, Pacific Northwest National Laboratory
Robert Sugar, University of California, Santa Barbara
Francis Sullivan, IDA/CCS

Theresa Windus, Pacific Northwest National Laboratory
Paul Woodward, University of Minnesota

B.8. Procurement, Accessibility, and Cost of Ownership

B.8.1. Charter

Explore the principal factors affecting acquisition and operation of HEC systems through the end of this decade. Identify those improvements required in procurement methods and means of user allocation and access. Determine the major factors that contribute to the cost of ownership of the HEC system over its lifetime. Identify impact of procurement strategy including benchmarks on sustained availability of systems

Questions:

- Evaluate the implications of the virtuous infrastructure cycle—i.e., the relationship among the advanced procurement development and deployment for shaping research, development, and procurement of HEC systems.
- Provide information about total cost of ownership beyond procurement cost, including space, maintenance, utilities, upgradeability, etc.
- Provide information about how the federal government can improve the processes of procuring and providing access to HEC systems and tools.

B.8.2. Participants

Frank Thames, NASA (*chair*)
Jim Kasdorf, Pittsburgh Supercomputing Center (*vice chair*)

Eugene Bal, Maui High Performance Computing Center
Rene Copeland, Platform Computing Federal, Inc.
Candace Culhane, National Security Agency
Charles Hayes, HCS
Cray Henry, DOD High-Performance Computing Modernization Office
Christopher Jehn, Cray
Sander Lee, DOE/NNSA
Matt Leininger, Sandia National Laboratories
Paul Muzio, Network Computing Service
Graciela Narcho, National Science Foundation
Per Nyberg, Cray
Thomas Page, National Security Agency
Steven Perry, Cray, Inc.
Mark Seager, Lawrence Livermore National Laboratory
Charles Slocomb, Los Alamos National Laboratory
Dale Spangenberg, National Institute of Science and Technology
Scott Studham, Pacific Northwest National Laboratory
James Tomkins, Sandia National Laboratories
William Turnbull, NOAA
Gary Walter, Environmental Protection Agency
W. Phil Webster, NASA
Gary Wohl, National Weather Service
Thomas Zacharia, Oak Ridge National Laboratory

APPENDIX C. LIST OF ATTENDEES

Kamal Abdali
National Science Foundation

Stanley C. Ahalt
Ohio State University/Ohio Supercomputer Center

Steven F. Ashby
Lawrence Livermore National Lab

David A. Bader
University of New Mexico

David H. Bailey
Lawrence Berkeley National Lab

Eugene Bal
Maui High Performance Computing Center

Robert A. Ballance
University of New Mexico

Donald B. Batchelor
Oak Ridge National Lab

Fernand D. Bedard
NSA

Herbert S. Bennett
N.I.S.T.

Andrew Bernat
Computing Research Association

David E. Bernholdt
Oak Ridge National Lab

Gyan V. Bhanot
IBM Research

Bryan A. Biegel
NASA Ames Research Center

Rupak Biswas
NASA Ames Research Center

Maurice L. Blackmon
NCAR

Patrick J. Bohrer
IBM

Ivo Bolsens
XILINX

Jon M. Boyens
U.S. Dept. of Commerce

Ron B. Brightwell
Sandia National Labs

Robert W. Brodersen
University of California, Berkeley

Walter Brooks
NASA Ames Research Center

Jeffrey S. Brown
LANL

Duncan A. Buell
University of South Carolina

David Callahan
Cray, Inc.

Kirk Cameron
University of South Carolina

William W. Carlson
IDA Center for Computing Sciences

Siddhartha Chatterjee
IBM T.J. Watson Research Center

Yolanda L. Comedy
IBM, Governmental Programs

Rene G. Copeland
Platform Computing Federal, Inc.

Thomas H. Cormen
Dartmouth College

George R. Cotter
National Security Agency

Deborah L. Crawford
National Science Foundation

Loring G. Craymer
Jet Propulsion Lab

Candace S. Culhane
National Security Agency

Tamara L. Dahlgren
Lawrence Livermore National Lab

William J. Dally
Stanford University

James Davenport
Brookhaven National Lab

Bronis R. de Supinski
Lawrence Livermore National Lab, CASC

Erik P. DeBenedictis
Sandia National Labs

Martin M. Deneroff
SGI

Yuefan Deng
Stony Brook University

Jack B. Dennis
MIT Lab for Computer Science

Carleton DeTar
University of Utah, Physics Dept.

Judith E. Devaney
NIST

David A. Dixon
Pacific Northwest National Lab

Jack J. Dongarra
University of Tennessee

Don Dossa
Lawrence Livermore National Lab

Thom H. Dunning Jr.
UT/ORNL Joint Institute for Computational Science

Douglas L. Dwoyer
NASA Langley Research Center

Mootaz Elnozahy
IBM Austin Research Lab

Wael R. Elwasif
Oak Ridge National Lab

Thomas M. Engel
NCAR

Thomas G.W. Epperly
Lawrence Livermore National Lab

William J. Feiereisen
Los Alamos National Lab/CCS Division

Stuart I. Feldman
IBM

Wesley M. Felter
IBM Austin Research Lab

James R. Fischer
NASA/Goddard Space Flight Ctr.

Susan Fratkin
CASC

Njema J. Frazier
U.S. Dept of Energy/NNSA

Peter A. Freeman
National Science Foundation

David A. Fuller
Joint National Integration Ctr.

Dennis Gannon
Indiana University, Computer Science Dept.

Guang R. Gao
Delaware Biotechnology Institute, University of Delaware

Ahmed Gheith
IBM

Eng Lim Goh
Silicon Graphics, Inc. (SGI)

Brent C. Gorda
Lawrence Berkeley National Lab

Howard Gordon
NSA

Steven A. Gottlieb
Indiana University

Robert B. Graybill
DARPA

Gary A. Grider
Los Alamos National Lab

John Grosh
DoD/OSD

Brian D. Gross
U.S. DOC/NOAA/GFDL, Princeton University

Maciej S. Gutowski
Pacific Northwest National Lab

John H. Hall
Los Alamos National Lab

Guy S. Hammer
Missile Defense Agency

Leslie B. Hart
NOAA/FSL

Charles W. Hayes
HCS

Michael J. Henesey
SRC Computers, Inc.

Cray J. Henry
DoD High Performance Computing Modernization Office

Daryl W. Hess
National Science Foundation MPS/DMR

Richard L. Hilderbrandt
ACIR/CISE/NSF

Phil Hilliard
National Academies, CSTB

Richard S. Hirsh
ACIR/CISE/NSF

Daniel A. Hitchcock
U.S. Dept. of Energy

Thuc T. Hoang
DOE/NNSA

Adolfy Hoisie
Los Alamos National Lab

Sally E. Howe
National Coordination Office, ITRD

Gary D. Hughes
National Security Agency

Curtis L. Janssen
Sandia National Labs

Stephen C. Jardin
Princeton Plasma Physics Lab

Christopher Jehn
Cray Inc.

Gary M. Johnson
U.S. Dept of Energy, Office of Science

Frederick C. Johnson
Dept. of Energy/Office of Science

Tina M. Kaarsberg
U.S. House of Representatives

David K. Kahaner
Asian Technology Information Program (ATIP)

Laxmikant V. Kale
University of Illinois - Urbana Champaign

James R. Kasdorf
Pittsburgh Supercomputing Center

Jeremy V. Kepner
MIT Lincoln Lab

Kirk T. Kern
Silicon Graphics Inc.

Frankie D. King
NCO/ITRD

Charles H. Koelbel
Rice University

David Koester
MITRE

Peter M. Kogge
University of Notre Dame

William T.C. Kramer
NERSC/LBNL

Norman H. Kreisman
Dept. of Energy

Gary K. Kumfert
CASC/LLNL

Alan J. Laub
SciDAC, DOE Office of Science

Sander L. Lee
DOE/NNSA

Charles R. Lefurgy
IBM

Matt L. Leininger
Sandia National Labs

John M. Levesque
Cray, Inc.

Greg B. Lindahl
Key Research, Inc.

Peter M. Lyster
National Institutes of Health

Arthur B. Maccabe
University of New Mexico, HPC

Kevin P. Martin
Georgia Institute of Technology

Sally A. McKee
Cornell University

Tyce T. McLarty
Lawrence Livermore National Lab

Thomas M. McWilliams
Key Research

Stephen P. Meacham
National Science Foundation

Bob Meisner
U.S. Dept. of Energy/NNSA

Juan C. Meza
Lawrence Berkeley National Lab

Grant Miller
NCO/Noesis

Ronald G. Minnich
Los Alamos National Lab

Richard L. Moore
UCSD - San Diego Supercomputer Ctr.

Virginia Moore
NCO/ITRD

Frank Mueller
North Carolina State University

Jose L. Munoz
DOE/NNSA

Paul C. Muzio
Network Computing Service, Inc.

Graciela L. Narcho
National Science Foundation

David B. Nelson
National Coordination Office, ITRD

Jeff A. Nichols
Oak Ridge National Lab

Jarek Nieplocha
Pacific Northwest National Lab

Michael L. Norman
UCSD

Per E. Nyberg
Cray, Inc.

George Ostrouchov
Oak Ridge National Lab

Thomas W. Page
National Security Agency

Dhabaleswar K. Panda
Ohio State University, Dept. of Computer Science

Joel R. Parriott
OMB

Steven S. Perry
Cray Inc.

Keshav K. Pingali
Cornell University

Neil D. Pundit
Sandia National Labs

Karthick Rajamani
IBM Austin Research Lab

Daniel A. Reed
University of Illinois/NCSA

Steven P. Reinhardt
SGI

Joseph D. Retzer
US EPA, Office of Environmental Information

Matthew Rosing
Peak Five

Robert D. Ryne
Lawrence Berkeley National Lab

Subhash Saini
NASA Ames Research Center

Nagiza F. Samatova
Oak Ridge National Lab

Ahmed H. Sameh
Purdue University

Barry I. Schneider
National Science Foundation, Physics Division

Rob Schreiber
Hewlett Packard

Steven L. Scott
Cray, Inc.

Mark K. Seager
Lawrence Livermore National Lab

Keith A. Shields
Cray Inc.

Suresh N. Shukla
The Boeing Company

Anthony Skjellum
MPI Software Technology, Inc.

Charles A. Slocomb
Los Alamos National Lab

Burton J. Smith
Cray, Inc.

Allan Edward Snavely
SDSC

Lawrence Snyder
University of Washington, CSE

Dale Spangenberg
NIST/OD/CIO

John W. Spargo
Northrop-Grumman

Vason P. Srini
UC Berkeley

Thomas L. Sterling
Caltech

Rick L. Stevens
Argonne National Lab, MCS Division

G. Malcolm Stocks
Oak Ridge National Lab

TP Straatsma
Pacific Northwest National Lab

Michael R. Strayer
ORNL

Scott Studham
Pacific Northwest National Lab

Robert L. Sugar
University of California, Santa Barbara

Frank C. Thames
NASA Langley Research Center

William Thigpen
NASA Ames Research Center

James L. Tomkins
Sandia National Labs

Jeroen Tromp
California Institute of Technology

William T. Turnbull
NOAA/HPCC Office

Augustus K. Uht
University of Rhode Island

Keith D. Underwood
Sandia National Labs

Sheila Vaidya
Lawrence Livermore National Lab

John R. van Rosendale
DOE, Office of Science

Alexander V. Veidenbaum
University of California, Irvine

Jeffrey S. Vetter
Lawrence Livermore National Lab

Uzi Vishkin
University of Maryland, IACS

Steven J. Wallach
Chiaro Networks

Gary L. Walter
U.S. EPA

Lee Ward
Sandia National Labs

John C. Wawrzynek
University of California, Berkeley

W. Phil Webster
NASA

Stephen R. Wheat
HPC Program Office, Intel Corporation

Barbara J. Wheatley
National Security Agency

Theresa L. Windus
Pacific Northwest National Lab

Gary M. Wohl
National Weather Service

Paul R. Woodward
University of Minnesota

Patrick H. Worley
Oak Ridge National Lab

Steven B. Yabusaki
Pacific Northwest National Lab

Thomas Zacharia
Oak Ridge National Lab

Asaph N. Zemach
Cray Inc.

Xiaodong Zhang
National Science Foundation

John P. Ziebarth
Los Alamos National Lab

Hans P. Zima
Jet Propulsion Lab